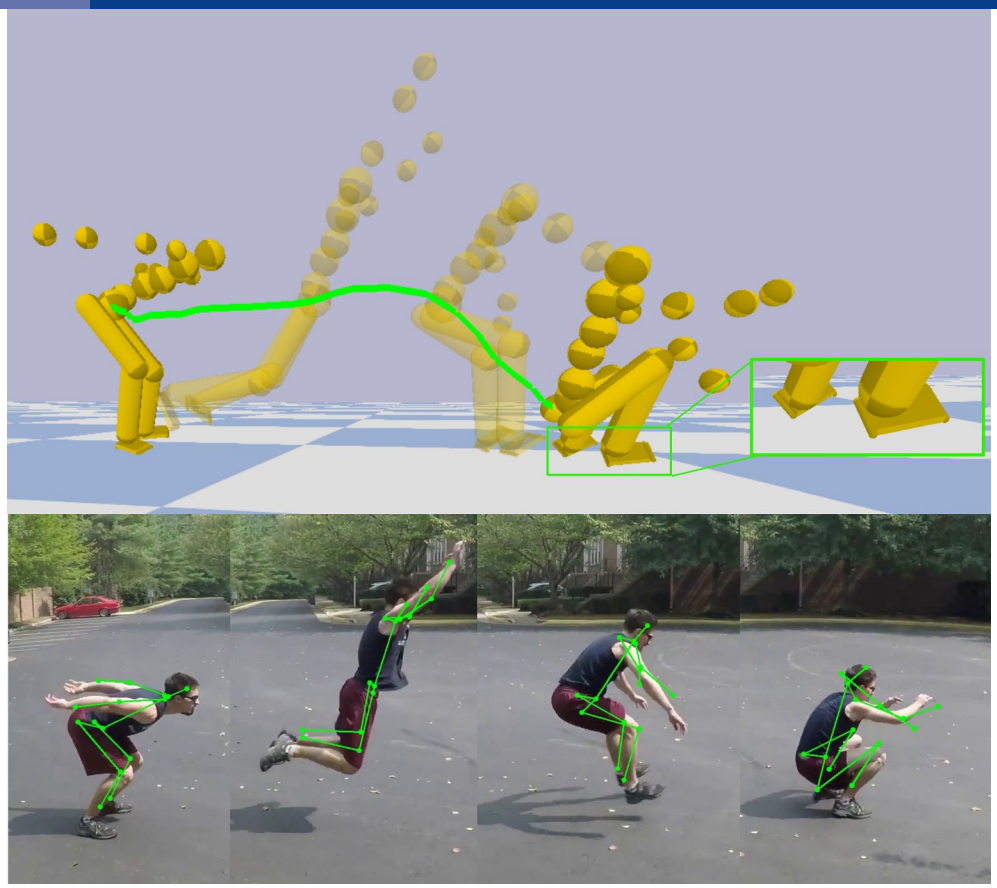
Organ der Gesellschaft für Informatik e.V.
und mit ihr assoziierter Organisationen

Band 44 • Heft 2 • April 2021

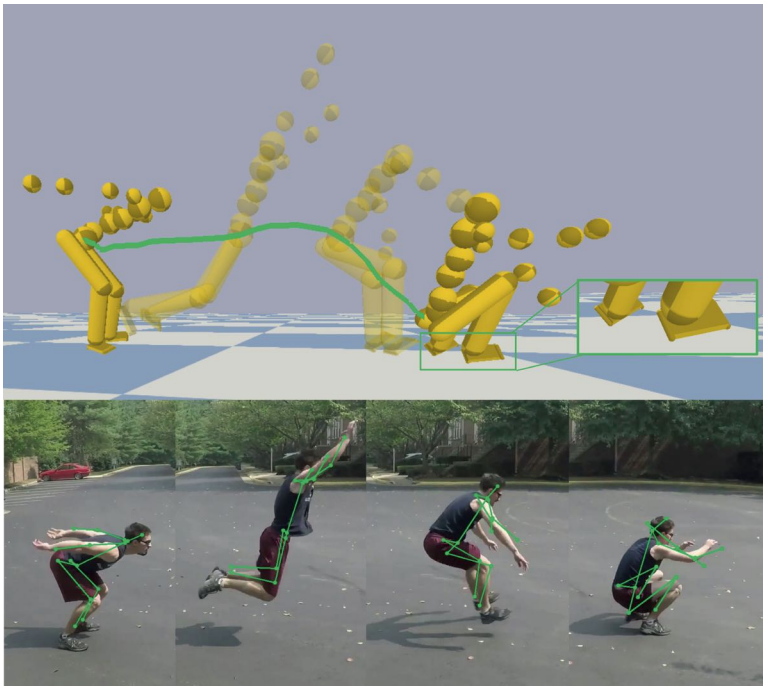
Informatik Spektrum



Springer

Software Engineering
E-ID-Gesetz und Datenschutz
Informatikunterricht in Deutschland – eine Übersicht

Informatik Spektrum
Organ der Gesellschaft für Informatik e. V. und mit ihr assoziierter Organisationen



Physikalisch plausible monokulare 3D-Bewegungserfassung

Unser neues Framework „PhysCap“ erfasst umfassende menschliche 3D-Bewegungen in einer physikalisch plausiblen Weise aus 2D-Videos in Echtzeit, automatisch und ohne den Einsatz von Markern. Die Abbildung zeigt einen Standweitsprung und unsere 3D-Rekonstruktionen.

Dank seiner Entwicklung auf der Basis physikalisch basierter Dynamik, kann unser Algorithmus komplexe menschliche 3D-Bewegungen, die in 2D aufgenommen wurden, wiederherstellen und Artefakte wie das Gleiten der Füße, das Durchdringen des Fußbodens, unnatürliche Körperneigung und Flackern vermeiden, die früheren monokularen Posenschätzungsmethoden Probleme bereiteten.

Quelle: Max-Planck-Institut für Informatik

Informatik Spektrum
Band 44 | Heft 2 | April 2021
Organ der Gesellschaft für Informatik e. V. und mit ihr assoziierter Organisationen

Springer

Informatik Spektrum

Organ der Gesellschaft für Informatik e. V. und mit ihr assoziierter Organisationen

Hauptaufgabe dieser Zeitschrift ist die Weiterbildung aller Informatiker durch Veröffentlichung aktueller, praktisch verwertbarer Informationen über technische und wissenschaftliche Fortschritte aus allen Bereichen der Informatik und ihrer Anwendungen. Dies soll erreicht werden durch Veröffentlichung von Übersichtsartikeln und einführenden Darstellungen sowie Berichten über Projekte und Fallstudien, die zukünftige Trends aufzeigen.

Es sollen damit unter anderem jene Leser angesprochen werden, die sich in neue Sachgebiete der Informatik einarbeiten, sich weiterbilden, sich einen Überblick verschaffen wollen, denen aber das Studium der Originalliteratur zu zeitraubend oder die Beschaffung solcher Veröffentlichungen nicht möglich ist. Damit kommt als Leser nicht nur der ausgebildete Informatikspezialist in Betracht, sondern vor allem der Praktiker, der aus seiner Tagesarbeit heraus Anschluss an die wissenschaftliche Entwicklung der Informatik sucht, aber auch der Studierende an einer Fachhochschule oder Universität, der sich Einblick in Aufgaben und Probleme der Praxis verschaffen möchte.

Durch Auswahl der Autoren und der Themen sowie durch Einflussnahme auf Inhalt und Darstellung – die Beiträge werden von mehreren Herausgebern referiert – soll erreicht werden, dass möglichst jeder Beitrag dem größten Teil der Leser verständlich und lesenswert erscheint. So soll diese Zeitschrift das gesamte Spektrum der Informatik umfassen, aber nicht in getrennte Sparten mit verschiedenen Leserkreisen zerfallen. Da die Informatik eine sich auch weiterhin stark entwickelnde anwendungsorientierte Wissenschaft ist, die ihre eigenen wissenschaftlichen und theoretischen Grundlagen zu einem großen Teil selbst entwickeln muss, will die Zeitschrift sich an den Problemen der Praxis orientieren, ohne die Aufgabe zu vergessen, ein solides wissenschaftliches Fundament zu erarbeiten. Zur Anwendungsorientierung gehört auch die Beschäftigung mit den Problemen der Auswirkung der Informatikanwendungen auf den Einzelnen, den Staat und die Gesellschaft sowie mit Fragen der Informatik-Berufe einschließlich der Ausbildungsrichtlinien und der Bedarfsschätzungen.

Vertrieb, Abonnement, Versand

Papierausgabe: ISSN 0170-6012
elektronische Ausgabe: ISSN 1432-122X
Erscheinungsweise: zweimonatlich

Den Bezugspreis können Sie beim Customer Service erfragen: customerservice@springernature.com. Die Lieferung der Zeitschrift läuft weiter, wenn sie nicht bis zum 30.9. eines Jahres abbestellt wird. Mitglieder der Gesellschaft für Informatik und der Schweizer Informatiker Gesellschaft erhalten die Zeitschrift im Rahmen ihrer Mitgliedschaft.

Bestellungen oder Rückfragen nimmt jede Buchhandlung oder der Verlag entgegen.
SpringerNature, Kundenservice Zeitschriften,
Tiergartenstr. 15, 69121 Heidelberg, Germany
Tel. +49-6221-345-0, Fax: +49-6221-345-4229,
e-mail: customerservice@springernature.com
Geschäftszeiten: Montag bis Freitag 8–20 h.

Bei Adressänderungen muss neben dem Titel der Zeitschrift die neue und die alte Adresse angegeben werden. Adressänderungen sollten mindestens 6 Wochen vor Gültigkeit gemeldet werden. Hinweis gemäß §4 Abs. 3 der Postdienst-Datenschutzverordnung: Bei Anschriftenänderung des Beziehers kann die Deutsche Post AG dem Verlag die neue Anschrift auch dann mitteilen, wenn kein Nachsendeauftrag gestellt ist. Hiergegen kann der Bezieher innerhalb von 14 Tagen nach Erscheinen dieses Heftes bei unserer Abonnementsbetreuung widersprechen.

Elektronische Version

springerlink.com

Hinweise für Autoren

http://springer.com/journal/00287

Impressum

Verlag:
Springer, Tiergartenstraße 17,
69121 Heidelberg

Redaktion:
Peter Pagel, Vanessa Keinert
Tel.: +49 611 787 8329
e-mail: Peter.Pagel@springer.com

Herstellung:
Madeleine Schnurr,
e-mail: Madeleine.Schnurr@springer.com

Redaktion GI-Mitteilungen:
Cornelia Winter
Gesellschaft f¨ur Informatik e.V. (GI)
Wissenschaftszentrum,
Ahrstraße 45, D-53175 Bonn,
Tel.: +49 228-302-145, Fax: +49 228-302-167,
Internet: http://www. gi.de,
e-mail: gs@gi.de

Wissenschaftliche Kommunikation:
Anzeigen: Eva Hanenberg
Abraham-Lincoln-Straße 46
65189 Wiesbaden
Tel.: +49 (0)611/78 78-226
Fax: +49 (0)611/78 78-430
eva.hanenberg@springer.com

Satz:
le-tex publishing services GmbH, Leipzig

Druck:
Printforce,
The Netherlands

springer.com

Informatik Spektrum (2021) 44:81
https://doi.org/10.1007/s00287-021-01350-2

EDITORIAL

Ist ja alles so schön bunt hier ...

Peter Pagel[1]

Angenommen: 10. März 2021

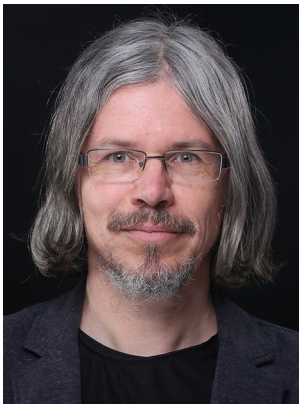
einigen ist es sicher schon aufgefallen, seit der vergangenen Ausgabe sind die Abbildungen im Informatik Spektrum farbig. Bislang waren diese schwarzweiß – ein Relikt aus der Historie dieser traditionsreichen Zeitschrift. Farbe ist bekanntlich mehr als Schmuck, viele Grafiken und Illustrationen sind leichter zu erfassen oder aussagekräftiger, wenn mehrere Farben und nicht nur Grauschattierungen verwendet werden – vielleicht ein Grund mehr, um über ein Abonnement des gedruckten Heftes nachzudenken?

Mit Heft 1/2021 hatten wir uns einem ungewöhnlichen Thema gewidmet, der digitalen Kunst. Uns und den Beteiligten hat das viel Spaß gemacht, wir hoffen, Ihnen ebenso. Sollten Sie Ideen oder konkrete Vorschläge für außergewöhnliche Inhalte mit IT-Bezug haben, hat die Redaktion dafür immer ein offenes Ohr. Wir freuen uns auf Nachrichten von Ihnen.

Das aktuelle Heft wendet sich wieder allgemeinen Themen zu. Die hier gesammelten Texte drehen sich um so unterschiedliche Themen wie den Informatikunterricht in Deutschland, Machine Intelligence, die Fernlehre zu Objektorientierter Programmierung sowie Software Engineering. Außerdem erklärt Gunter Dueck, weshalb ein digitaler Workaround keine Digitalisierung ist. Wir wünschen allen Lesern eine spannende Lektüre und viele neue Sichtweisen und Anregungen.

Bleiben Sie gesund!

Peter Pagel
Chefredakteur



Peter Pagel

✉ Peter Pagel
peter.pagel@springer.com

[1] Wiesbaden, Deutschland

Informatik Spektrum (2021) 44:82–94
https://doi.org/10.1007/s00287-020-01322-y

HAUPTBEITRAG

Software Engineering und Software-Engineering-Forschung im Zeitalter der Digitalisierung

Ein Beitrag über das aktuelle und zukünftige Selbstverständnis des Software Engineering

Michael Felderer[1] · Ralf Reussner[2] · Bernhard Rumpe[3]

Online publiziert: 13. November 2020

Zusammenfassung
Die Digitalisierung und der damit verbundene digitale Wandel durchdringen alle Lebensbereiche. Qualitativ hochwertige Software ist der zentrale Baustein und Treiber der Digitalisierung. Damit nimmt auch das ingenieurmäßige Erstellen von Software, das Software Engineering, eine zentrale Rolle im digitalen Wandel ein und ist dabei selbst großen Veränderungen unterworfen. Dieser Artikel versucht deshalb eine Standortbestimmung des Software Engineering und seiner Forschung im Zeitalter der Digitalisierung vorzunehmen.

> „Die Informatik verändert sich. Deshalb müssen auch die Informatiker Ihre Rolle in der Gesellschaft neu überdenken."
> von Ranga Yogeshwar 2018 auf der Informatik Tagung der GI.

Aktueller Stand und Beobachtungen

Die Digitalisierung wirkt sich nicht nur auf die Gesellschaft aus, sie verlangt auch eine Neubestimmung des Standorts der Informatik und der InformatikerInnen, wie sie der Wissenschaftsjournalist Yogeshwar in seinem obigen Zitat anregt. Da gerade sämtliche Aspekte der Digitalisierung auf Software beruhen, soll dieser Artikel der Versuch einer Neubestimmung insbesondere der Rolle des Software Engineering und seiner Forschung sein. Schon jetzt haben softwarebasierte Produkte, Systeme oder Dienstleistungen praktisch alle Lebensbereiche durchdrungen. Dadurch wird Software und damit das Software Engineering zum kritischen Baustein und zentralen Innovationstreiber der Digitalisierung in allen Lebensbereichen. Auch wissenschaftlich ergeben sich neue Chancen und Herausforderungen für das Software Engineering als treibende Disziplin bei der Entwicklung jedweder technischer Innovation. Gerade die Chancen dürfen allerdings auch nicht dem Wettbewerb um bibliometrische Zahlen als Selbstzweck geopfert werden.

Trends

Konkret können wir gegenwärtig wenigstens diese sieben Trends über Software und das Software Engineering beobachten:

1. Software spielt in vielen Produkten die Rolle des wesentlichen Innovationstreibers oder Produkte werden durch softwarebasierte Dienstleistungen ergänzt, um sich am Markt wesentlich zu differenzieren. Beispiele sind automatisiertes Fahren oder die Digitalisierung von Geschäftsprozessen.
2. Software integriert Geräte und Dienstleistungen und ist damit der Kern sogenannter Systems-of-Systems. Beispiele sind die Themen zusammengefasst unter „Industrie 4.0", das Smart Energy Grid oder moderne multimodale Mobilitätssysteme.
3. Software wird sogar noch zunehmend von Personen entwickelt, die dies nicht primär gelernt haben. Das liegt

✉ Michael Felderer
michael.felderer@uibk.ac.at

[1] Universität Innsbruck, Innsbruck, Österreich

[2] KIT Karlsruhe, Karlsruhe, Deutschland

[3] RWTH Aachen, Aachen, Deutschland

zum Teil an der Unterversorgung des Arbeitsmarkts und zum Teil an dem zur Entwicklung notwendigen Domänenwissen. Notwendiger- und glücklicherweise wird zumindest die Programmierung von Software zum Commodity, deren Möglichkeiten und Grundzüge allerdings jeder wenigstens ansatzweise verstehen sollte.

4. Software nutzt zunehmend KI-Techniken, um Funktionalitäten zu lernen, zu verbessern oder anzupassen. Dadurch wird die Überprüfung von Qualitätseigenschaften während der Entwicklungszeit eingeschränkt und manchmal sogar verhindert, da das eigentliche Verhalten der Software erst nach der Trainingsphase absehbar ist, und nicht nur durch bloße Analyse des Codes oder das klassische Testen von Anforderungen deduziert werden kann. Beispiele finden sich etwa in der Bild- und Sprachverarbeitung und damit auch in sicherheitskritischen Systemen wie autonomen Fahrzeugen. Obwohl dies zum Beispiel aufgrund der Nutzung von Datenbanken immer schon partiell so war, übernehmen nun die Daten eine stärkere Rolle bei der Definition des Verhaltens von Software. Zudem haben KI-Techniken (etwa durch neue Verfahren der Bild- und Spracherkennung) auch zu einer Erweiterung des Spektrums der Benutzerschnittstellen und damit der Bedeutung der User Experience beigetragen.
5. Auch für den Forschungsprozess im Software Engineering selbst sind Daten entscheidend, um Hypothesen empirisch zu prüfen. Allerdings birgt die beinahe beliebige Verfügbarkeit von Daten im Software Engineering auch Gefahren für den Forschungsprozess. Zu häufig werden Daten etwa aus Softwarerepositorien oder durch Umfragen generiert und anschließend publiziert ohne diese durch eine Theorie, eine genaue Kontextdefinition oder eine Vision zur Entwicklung einer Methode zu fundieren, wodurch der Nutzen dieser Arbeiten für die Disziplin des Software Engineering unklar bleibt.
6. Software greift immer normativer in Lebensrealitäten ein. Softwareprodukte haben bereits heute institutionellen Charakter: Sie regeln und machen Vorgaben, allerdings oft ohne die Legitimation der üblichen Institutionen. Sie werden daher in Zukunft notwendigerweise zunehmend von staatlicher Seite reguliert. Dazu gehört insbesondere der Schutz personenbezogener Daten als essenzielle Komponente des Schutzes der Personen selbst.
7. Der Umgang mit Software wirkt sich auch auf die Kognition und die Persönlichkeit der BenutzerInnen sowie auf die Gesellschaft als Ganzes aus. Diese Wirkungen sind heute zwar erkennbar, aber noch nicht wirklich abgeschätzt oder gar reflektiert bei der Entwicklung der Software.

Mit der Industrialisierung und der Vergesellschaftung von Software und Daten einhergehend ist auch zu beobachten, dass immer mehr Menschen aktiv in die Entwicklung von Software eingebunden sind, sei es durch die Bereitstellung von Anforderungen, das Testen von Systemen im Feld oder die Teilnahme an der Softwareentwicklung selbst und dass mit immer kürzeren Innovationszyklen auch immer kürzere Entwicklungszyklen einhergehen. Konkret wird Software zur Laufzeit durch nachladbare Komponenten und Apps stark veränderlich, erweiter- und anpassbar. Der Mensch ist als Datenquelle heute (mehr passiv) Teil von Softwareökosystemen. Aber auch zur aktiven Konfiguration (z. B. der vielen einzelnen Geräte im zukünftigen E-Home) werden immer mehr Menschen die Fähigkeit benötigen, rudimentär Software zu programmieren („Programming as a Commodity“). Zudem rückt der Mensch als User immer mehr in den Fokus der Entwicklung, um durch neue Interfaces wie Sprachein- und -ausgabe oder Augmented Reality die User Experience zu optimieren. Software Engineering muss sich also in Zukunft nicht nur mit der Entwicklung von Software beschäftigen, sondern auch mit deren hoch-adaptiver Konfigurierbarkeit, Benutzerakzeptanz und der Qualitätssicherung von konfigurierter und durch Daten trainierter Software.

Schon aus technischer Perspektive ergeben sich aus den obigen Beobachtungen neue Herausforderungen, die auch eine neue Standortbestimmung der nunmehr 50 Jahre alten Disziplin des Software Engineering [1–4] erfordern:

1. Software Engineering nimmt Regulationen und Werteverständnisse auf und versucht Verfahren, Algorithmen und deren Konfigurationsdaten sowie Architekturen zu entwickeln, um diese systematisch umzusetzen.
2. Software Engineering verpackt Softwareansätze so, dass auch Nicht-SoftwaretechnikerInnen sie nutzen können. Als Beispiel und Vorbild können Datenbanksysteme gelten, die auch von Nicht-InformatikerInnen sinnvoll eingesetzt werden können. Die Nutzbarmachung wesentlicher Algorithmen, insbesondere des maschinellen Lernens, ist für andere Disziplinen eine weitere Schnittstelle, zugegebenermaßen nicht nur, aber auch zu den Sozial- und Gesellschaftswissenschaften.
3. Software Engineering beschäftigt sich natürlich primär mit Software und den notwendigen Grundlagen ihrer Entwicklung. Aber auf beide Bereiche haben Menschen und Organisationen einen entscheidenden Einfluss. Die Entwicklung und Untersuchung von Software-Engineering-Artefakten bedarf deshalb die Einbeziehung des Entstehungs- und des Anwendungskontextes. Unter anderem deshalb ist die Anwendung von empirischen Forschungsmethoden und Theorien aus den Sozialwissenschaften wie Psychologie oder Soziologie notwendig, um menschliche und organisatorische Kontextfaktoren

geeignet in den Software-Engineering-Forschungsprozess weiter zu integrieren. Software Engineering hat in diesem Sinne nicht nur Züge von traditionellen Ingenieurswissenschaften wie Maschinenbau oder Elektrotechnik, sondern unterscheidet sich von diesen durch diesen zusätzlichen sozialwissenschaftlichen Charakter.

Konsequenzen für das Software Engineering

Daraus lassen sich einige Konsequenzen für Software Engineering ableiten:

1. Eine der wesentlichsten Konsequenzen aus diesen Beobachtungen ist, dass Software bzw. die Handlungen, Vorschläge und Entscheidungen, die Software trifft oder ausführt, in Zukunft besser nachvollziehbar sein müssen. Dies gilt insbesondere bei adaptiver Software, bei lernender Software und bei Software, die in dem oben genannten Maß durch EndnutzerInnen modifizierbar sein soll. Solche Software zu erstellen ist natürlich auch für Software Engineering eine Herausforderung, da sowohl die Form der Erklärung für NutzerInnen einfach und gut verstehbar sein muss, als auch zum Beispiel bei adaptiver und lernender Software Erklärungen nicht notwendigerweise vorab bereits definiert und fixiert werden können.
2. Software Engineering drängt sich vermehrt in die Rolle des umfassenderen Systems Engineering. Durch die zunehmend dominierende Rolle von Software in technischen Systemen verändert sich der Zusammenhang von Software und Systems Engineering. Konnte man Software Engineering bisher als einen Teil des Systems Engineering betrachten, so bietet heute Software Engineering selbst oft die grundlegenden Methoden für die Entwicklung softwareintensiver technischer Systeme. Software Engineering entwickelt sich deshalb von einer „Nebenbeschäftigung“ des Systems Engineers zum primären Treiber der Entwicklung. Daraus lassen sich einige Konsequenzen ableiten, die vor allem Entwicklungsprozesse betreffen. Immer noch beißen sich der funktionsorientierte Softwareentwicklungsprozess mit dem vor allem geometrisch dekomponierenden Systementwicklungsprozess. Viele Methoden zur Systematisierung und Automatisierung der Software-Engineering-Aktivitäten bei Dekomposition, Synthese, Qualitätssicherung, aber auch im Management von Evolution und Varianten müssen noch in den Systementwicklungsprozess übertragen werden. Erst dann lassen sich auch die von vielen gewünschten agilen Vorgehensmethoden stärker und gewinnbringend in die Systementwicklung einbringen.
3. Software Engineering war es immer und wird es noch stärker werden: eine Integrationsdisziplin. Durch Software werden einzelne Produkte und Dienstleistungen vernetzt. Damit ergibt sich durch Software oft erst die Möglichkeit, innovative Szenarien umzusetzen, z. B. für Energieversorgung mit regenerativen Energieformen, Mobilität in der Kombination verschiedener Verkehrsträger oder zur komplexen Wertschöpfungsketten vernetzte Produktionsanlagen. Dadurch kommt den SoftwareingenieurInnen aber auch eine wesentliche Rolle in der „Findung“ oder sogar „Erfindung“ und Definition der eigentlichen Funktionalität des Systems-of-Systems zu. Während sich durch die Vernetzung die Produkte immer mehr integrieren, differenzieren sich die Rollen und Fähigkeiten der an einem Produkt beteiligten EntwicklerInnen immer weiter aus. Dies gilt sowohl für Produkte, als auch für die immer stärker vernetzte und digitalisierte Welt der Produktion. Dabei nicht zu vernachlässigen sind auch die immer komplexeren Softwarewerkzeuge, die zur Entwicklung selbst eingesetzt werden und aufgrund der Vernetzung ihrerseits die Verbindung zum ausgelieferten, aber beobachtbaren und aktualisierbaren Produkt halten werden.
4. Software wird datengetrieben adaptiv. Durch den Einsatz von Verfahren des maschinellen Lernens zur Erbringung von Softwarefunktionalität kann diese häufig nicht mehr zur Entwicklungszeit überprüft werden, da die Funktionalität nicht nur durch den Code, sondern auch durch (Trainings-)Daten festgelegt wird. Hier müssen wir verstehen lernen, unter welchen Bedingungen und inwieweit ein zu bauendes System noch zur Entwicklungszeit absicherbar ist, unter welchen Bedingungen eine Laufzeitüberwachung durch sogenannte „Monitore“ möglich ist, wann eine A-posteriori-Analyse von Fehlern noch zulässig ist oder auch wann ein System nicht gebaut werden darf, da es nicht mehr absicherbar ist. Und dies gilt sowohl dann, wenn das Training selbst bereits in der Entwicklungszeit abgeschlossen wird, als auch dann, wenn das System in Betrieb weiter lernt („life-long learning“).
5. Software muss normen- und gesetzessicher gebaut werden. Durch die notwendige zunehmende Regulierung softwarebasierter Produkte und Dienstleistungen gerade im Bereich des Datenschutzes und der Personensicherheit ergeben sich neue Anforderungen an die Software und ihre Entwicklungsmethodik. Bisher können Anforderungen wie der Schutz personenbezogener Daten gemäß sich wechselnder Einverständniserklärungen für verschiedene sich ändernde Arten der Datennutzung über den gesamten Lebenszyklus der Software nicht systematisch umgesetzt geschweige denn nachgewiesen werden.

Gemeinsam ist all diesen Herausforderungen, dass zum einen dadurch die Komplexität der Software weiter massiv ansteigen wird, zum anderen, dass immer mehr Menschen in verschiedenen Rollen mit Software und ihrer Entwick-

lung zu tun haben werden und von ihrer Funktionalität und von ihren Entscheidungen noch stärker als bisher abhängen werden. Insbesondere ist eine oben genannte Konsequenz daraus, dass auch der Bedarf an Nachvollziehbarkeit und Erklärbarkeit von Software steigen wird. Nur so wird auch die notwendige Akzeptanz beim Nutzer, bei Organisationen und in der Gesellschaft insgesamt erreichbar sein. Auch ist klar, dass es noch weniger als heute Personen geben wird, die ein Gesamtverständnis – oder auch nur einen detaillierten Überblick – haben werden. Die Konsistenzhaltung der verschiedenen Sichten auf ein System wird also immer wichtiger.

Weiterentwicklung des Software Engineering

Im Zuge des Jubiläums „50 Jahre Software Engineering" [1] – seit der NATO Konferenz 1968 – gibt es mehrere Diskussionen über deren Essenz und weitere Entwicklung des Software Engineering als Disziplin. Basierend auf der hohen praktischen Relevanz von Software-Engineering-Lösungen schlagen Basili und Briand [2] eine stärkere Kontextorientierung der Software-Engineering-Forschung vor, da die Anwendbarkeit und Skalierbarkeit von Software-Engineering-Lösungen zentral von Kontextfaktoren wie dem Menschen (etwa seines Persönlichkeitstyps und Wissenshintergrunds), der Organisation (etwa Kosten- und Zeitrestriktionen) oder der Domäne (wie etwa der Kritikalität oder der geforderten Compliance zu Standards) abhängt. In ihrem kontextgetriebenen Forschungsparadigma für das Software Engineering schlagen sie demnach vor, nicht mehr weiter primär top-down allgemeine Ansätze ohne Berücksichtigung von Kontextfaktoren zu entwickeln, sondern bottom-up in Kollaboration mit Praktikern innovative Lösungen zu entwickeln und dabei Zusammenhänge zu erkennen, Theorien zu entwickeln und wissenschaftlich vor allem durch den Einsatz empirischer Methoden zu evaluieren. Damit in Verbindung stehend hebt Broy [3] den integrativen Charakter des Software Engineering für alle etablierten Ingenieursdisziplinen hervor sowie die Notwendigkeit die Software-Engineering-Ausbildung aufgrund der strategischen und gesellschaftlichen Bedeutung von Software dahingehend zu erweitern, dass SoftwareingenieurInnen kompetent Entscheidungen von strategischer Tragweite treffen können. Booch [4] sieht Abstraktion, Separation of Concerns, Verteilung von Verantwortlichkeiten sowie Einfachheit zum Management von Komplexität als zentrale Fundamente des Software Engineering, die es zukünftig etwa auf KI-Systeme anzuwenden gilt, um Antworten auf zentrale Fragen in diesen Systemen, wie etwa nach ihrem Lebenszyklus, ihre Qualitätssicherung oder ihre Konfiguration, zufriedenstellend beantworten zu können.

Wir denken, die Weiterentwicklung des Software Engineering und insbesondere die integrative Anwendung des Software Engineering in Kombination mit anderen Disziplinen ist in den nächsten Jahren essenziell. In den letzten 50 Jahren hat sich innerhalb des Software Engineering Wesentliches getan. Eine solide und große Sammlung an Methoden erlaubt uns, punktgenau Agilität mit der konstruktiven Entwicklung und Synthese der Software und der analytischen Qualitätssicherung zu verzahnen. Requirements Engineering ist seinem Wesen nach schon immer eine hochgradig an Innovationen interessierte Methodik. Softwarearchitektur erlaubt es, funktionsorientiert, aufgabenorientiert, orientiert an den physikalischen Gegebenheiten oder datenorientiert komplexe Herausforderungen zu zerlegen, zu organisieren und in robuste Lösungen umzusetzen. Modellbildung hat als Wesenszug zur Abstraktion und Darstellung der funktionalen Essenz eines Systems in der Softwareentwicklung unter anderem durch die Standards UML und SysML, aber auch durch domänenspezifische Sprachentwicklungen Einzug gehalten. Varianten- und Versionsmanagement befinden sich genauso in der Werkzeugkiste der SoftwareingenieurInnen wie Simulations-, Testautomatisierungs- und Analysewerkzeuge zur Qualitätssicherung sowie Messwerkzeuge verschiedenster Arten.

Eine der großen Herausforderungen für Software-Engineering-Forschungsergebnisse selbst bleibt allerdings ihre praktikable Umsetzbarkeit durch Werkzeuge. Werkzeuge entstehen häufig als Nebenprodukt wissenschaftlicher Forschung oder werden durch eine gemeinsame Community als Open-Source-Werkzeuge so weit vorangetrieben, wie es dieser Community möglich ist. Werkzeuge sind heute hochkomplex und benötigen sowohl einen guten Bedienungskomfort, als auch die relevante Funktionssicherheit und müssen auf große Projekte skalieren.

Was hat Software Engineering auch für andere Disziplinen zu bieten?

Software Engineering ist zunächst wie jede Ingenieursdisziplin eine konstruktive Wissenschaft. In den 50 Jahren seit Bestehen der Disziplin wurde zunehmend gelernt, welche Konsequenzen sich aus der Besonderheit von Software ableiten, nicht materiell zu sein. Aktuell ist Software dabei, den Alltag von Individuen, Organisationen und letztlich der Gesellschaft in jeder Hinsicht so schnell wie kaum eine andere Disziplin zuvor zu verändern. Nahezu alle Disziplinen und Domänen sind in Begriff, sich zu digitalisieren. Für eine Standortbestimmung setzen wir deshalb das Software Engineering sowohl zu den anderen Ingenieurswissenschaften als auch den Sozialwissenschaften in Bezug.

Bezug zu anderen Ingenieurswissenschaften

Wie einleitend, beschrieben wird der Bezug zu anderen Ingenieurswissenschaften gegenwärtig geprägt durch

a. den steigenden Anteil von Software in vielen technischen Systemen,
b. der Vernetzung dieser Systeme zu Systems-of-Systems durch Software,
c. dem hohen Anteil an Software beim Innovationsgrad technischer Systeme sowie
d. dem antizipierten Nutzen, wenn spezifische softwaretechnische Vorgehensweisen auch im Systems Engineering angewandt werden.

Letztlich bedingen diese Faktoren einander, sodass wir sie gemeinsam mit dem letzten beginnend ausführen.

Im Vergleich zu den Artefakten anderer Ingenieursdisziplinen ermöglicht die Eigenschaft von Software, nicht materiell zu sein, einen größeren Entwurfsraum, der kaum durch physische Materialeigenschaften eingeschränkt ist. Werkzeuge und Methoden sind unter dem Stichwort Agilität in den letzten Jahren intensiv weiterentwickelt worden, sodass schnelle Feedback- und Innovationszyklen in der Software üblich sind. Seit etwa 2000 wissen wir, dass die Time-to-Market im Internetzeitalter noch viel wichtiger geworden ist und im Internet „ein Jahr nur drei Monate“ dauert. Negative Auswüchse des schnellen Entwickelns, wie etwa der Bananensoftware, die erst beim Kunden reift, sind durch methodische Maßnahmen, wie etwa aktive Beta-Tester, testgetriebene Entwicklung, frühe Reviews (z. B. auch Pair-Programming) oder kontinuierliches Monitoring des Kundenverhaltens und damit auch der Kundenzufriedenheit abgefangen worden.

Wenn Software selbst keine physikalische Manifestation hat, ist ihre „Produktion“ reduziert auf den Build-Prozess, das Deployment und die Konfiguration. Da diese Vorgänge weitgehend automatisierbar sind, ist vor allem der Entwicklungsprozess, um den sich traditionell ja das Software Engineering kümmert, der limitierende Faktor. Dabei spielen die einzelnen entwickelnden Menschen, die Teamstruktur und -zusammensetzung sowie die Randbedingungen durch die Organisation eine größere Rolle als bei anderen Ingenieurswissenschaften. Sie prägen den Prozess der Produktentwicklung erheblich und beeinflussen damit letztlich auch Qualitätseigenschaften des Produkts selbst intensiv. Diese menschlichen und organisatorischen Faktoren sind deshalb ebenfalls ein wichtiger Einflussfaktor und damit Untersuchungsgegenstand des Software Engineering. Darüber hinaus bietet das Software Engineering als Disziplin eine Fülle an Techniken und Methoden, welche die Effizienz und Effektivität der menschlichen Interaktion im Kontext der Softwareentwicklung optimieren.

Dadurch ergeben sich für die Entwicklung softwareintensiver technischer Systeme eine Reihe von Möglichkeiten, Einsichten aus dem Software Engineering zu übertragen. Dazu gehören zum Beispiel Vorgehensmodelle, die nicht mehr von der Vorausplanbarkeit eines ganzen Systems ausgehen, sondern eine dezentrale Entwicklung von Systemen erlauben, die lose gekoppelt sind und keinem gemeinsamen Entwicklungsprozess unterliegen, obwohl sie Systemkomponenten und -schnittstellen teilen. Dazu gehören zum Beispiel auch agile Verfahren zur Unterstützung schneller Innovationszyklen. Da Software letztlich auch ein Modell der Realität mehr oder weniger explizit und ausführlich in sich trägt, sind Verfahren der Modellierung aus dem Software Engineering, wie zum Beispiel die UML, SysML oder Meta-Modellierung, sowie die Synthese, Analyse und Transformation von Modellen von besonderem Interesse für das Systems Engineering, zumal der Umgang mit Modellen und deren Analyse und Simulation ohnehin zum bestehenden Ingenieursvorgehen passt. In der Praxis zeigt sich aber, dass die Modelle der Software Engineering und des Systems Engineering bzw. die zugrunde liegenden Paradigmen nicht einfach vereint werden können und daher noch viel grundlegende Forschung notwendig ist.

Besonders sichtbar ist die Nutzung von Modellierungstechniken, wenn diese nicht nur zur Entwicklung, sondern auch während des Betriebs der Software im Einsatz sind. Traditionell sind das zum Beispiel explizite Datenmodelle oder Geschäftsprozessmodelle, die manchmal in Grenzen, aber öfter auch flexibel anpassbar sein sollen. In der Produktwelt sind das digitale Twins, also rein virtuelle, durch Modelle und Simulationen betriebene „Zwillinge“ physischer Artefakte, die auch Meta-Information enthalten und durch prädiktive Techniken zur Vorhersage von Eigenschaften, aber insbesondere zur Kontrolle und Steuerung ihrer physischen Zwillinge eingesetzt werden können. Die gemeinsame, weil eng verzahnte Entwicklung des physischen Produkts und seines virtuellen Zwillings erfordert neue Techniken. Dies ist auch deshalb notwendig, weil auch viele physische Produkte mit der Zeit ergänzt, erweitert, aktualisiert oder durch Ersatzteile modifiziert werden und ein digitaler Zwilling dies reflektieren und idealerweise sogar antizipieren muss. Hier wird das bestehende Verständnis von „Digital Engineering“ substanziell erweitert, indem nicht „nur“ bestehende Entwicklungsprozesse durch Software unterstützt werden, sondern nun auch softwaretechnische Methoden neue Entwicklungsprozesse ermöglichen. Eine Vision wäre die integrierte Modellierung und Analyse in einem gemeinsamen Entwurfsraum, im dem Software, aber auch physische Artefakte methodisch gemeinsam behandelt werden. Natürlich müssen diese Verfahren systematisch mit Fragen der unterschiedlichen Lebenszyklen von Software und physischen Artefakten umgehen und dabei ihre Konsistenz sicherstellen.

Bezug zu den Sozialwissenschaften

Es ist wie immer im Software Engineering: Die Einführung neuer Systeme verändert die Verhaltensweisen und Aktivitäten (Geschäftsmodelle und -prozesse, Support, etc.) der NutzerInnen. Das muss so sein, denn sonst wäre die Software sinnlos. Die zurzeit massiv voranschreitende Digitalisierung wird daher starke Konsequenzen für Individuen, Organisationen, Unternehmen und die Gesellschaft haben, die durchaus mit denen der Industrialisierung im 19. Jahrhundert vergleichbar sind. Offensichtlich hat sich das Kommunikationsverhalten bereits massiv verändert. Untersuchungen zeigen aber auch Änderungen der menschlichen Kognition durch die allgegenwärtige Nutzbarkeit digitaler Geräte oder der Arbeitsweisen von Teams. Die Auswirkungen neu entstehender Geschäftsmodelle und ihrer gesellschaftlichen Auswirkungen können im sozialen und im Arbeitsleben immens sein, wenn die Wertschöpfung durch Automatisierung und dem damit auch verbundenen Wegfall traditioneller Arbeitsplätze zugunsten neu geschaffener digitaler Arbeitsplätze einhergeht. Dazu kommen Geschäftsmodelle, die aus personenbezogenen Daten Werte schöpfen und Social-Media-Anwendungen, die durch das permanente Feedback zu den Handlungen einzelner auch grundlegende Rechte wie Privatheit oder Gleichbehandlung zumindest neu austarieren.

Die Erforschung dieser Effekte auf Einzelne, Organisationen und die Gesellschaft insgesamt ist in den Sozialwissenschaften als Technologiefolgenabschätzung bereits im Gange und beeinflusst auch das Software Engineering.

Daher gibt es mehrere Schnittstellen des Software Engineering zu den Sozialwissenschaften, die auch deshalb relevant sind, weil viele Möglichkeiten der Digitalisierung und die daraus folgenden Verwendungsformen aktuell noch nicht umgesetzt sind, sondern höchstens angedacht werden. Dazu gehören sowohl Techniken der Speicherung und Verarbeitung personenbezogener Daten im Bereich Data Science als natürlich auch die Methoden der künstlichen Intelligenz, die im aktuellen Hype-Zyklus einige weitere Probleme lösen wird, aber auch dieses Mal alle nicht Probleme, zu deren Lösung Intelligenz notwendig ist, in den Griff bekommen dürfte. Software Engineering steht demnach zu den Sozialwissenschaften in folgenden Wechselwirkungen:

1. Software Engineering kann Szenarien liefern, die verständlich machen, welche Softwarefunktionalitäten in der nächsten Zukunft realisierbar sind oder realisierbar werden können. Hier sind durchaus auch sowohl Szenarien interessant, die das Wohl einzelner oder das Gesamtwohl der Menschen verbessern, aber auch Szenarien beschreibbar, die nach dem aktuellen Werteverständnis zwar möglich, aber nicht wünschenswert sind. Dadurch können Untersuchungen über die langfristigen Konsequenzen und Diskussionen über das, was gesellschaftlich wünschenswert ist, ausgelöst und faktenorientiert geführt werden.
2. Software Engineering nimmt Regulationen und Werteverständnisse auf und versucht, Verfahren, Algorithmen und Architekturen zu entwickeln, um diese systematisch umzusetzen. Dieser Punkt ist gewissermaßen das Gegenstück zum vorherigen Punkt, wo das Software Engineering Szenarien liefert; hier nimmt es solche von den Sozialwissenschaften auf.
3. Software Engineering unterstützt Softwareentwicklungsansätze mit Methoden und Werkzeugen, sodass auch Nicht-SoftwaretechnikerInnen sie nutzen können. Als Beispiel können die bereits genannten Datenbanksysteme gelten oder auch das Enduser-Computing, das in verschiedensten Formen weitere Verbreitung findet. Dazu gehören einfache und intuitive Konfigurationssprachen, wie sie etwa im Bereich der Home-Automatisierung angeboten werden, oder auch Wizards für bestimmte partikulare Probleme. Darüber hinaus trägt das Software Engineering auch zur Nutzbarmachung wesentlicher Algorithmen bei, insbesondere im Bereich des maschinellen Lernens von Zusammenhängen aus Datenmengen, wodurch den Sozialwissenschaften neue quantitative Methoden effektiv zur Verfügung stehen.
4. Mit der Durchdringung aller Lebensbereiche mit softwarebasierten Produkten und Dienstleistungen rückt der Mensch als Nutzer, dessen User Experience es zu optimieren gilt, immer mehr in den Fokus der Entwicklung. Solche Usability-Aspekte können nur mit sozialwissenschaftlichen Methoden, insbesondere aus der Psychologie, adäquat adressiert werden. Die Bedeutung der User Experience für den Erfolg von softwarebasierten Lösungen hat zusätzlich durch eine Vielzahl neuer Benutzerschnittstellen wie etwa für die Sprachein- und -ausgabe oder Augmented bzw. Virtual Reality stark zugenommen.
5. Wie bereits besprochen haben Menschen und Organisationen einen entscheidenden Einfluss auf das Software Engineering. Die Entwicklung und Untersuchung von Software-Engineering-Artefakten bedarf deshalb der Anwendung von empirischen Forschungsmethoden und Theorien aus den Sozialwissenschaften wie der Psychologie oder Soziologie, um menschliche und organisatorische Kontextfaktoren geeignet in den Software Engineering Forschungsprozess integrieren zu können. Software Engineering hat in diesem Sinne nicht nur Züge von traditionellen Ingenieurswissenschaften wie Maschinenbau oder Elektrotechnik, sondern unterscheidet sich von diesen durch diesen ausgeprägteren sozialwissenschaftlichen Charakter.

Eigentlich war es schon immer eine Aufgabe des Software Engineering, sicherzustellen, dass Software nachvollziehbar ist. Allerdings steigt die Komplexität von Software, die aus immer umfangreicheren Technologie-Stacks sowie aus immer mehr und stärker vernetzten Anwendungskomponenten entsteht, weiter deutlich an. Es werden immer häufiger selbstlernende Komponenten, aber auch traditionell programmierte intelligentere („smarte“) Komponenten zur Kontrolle immer kritischerer und mit immer mehr Sensorik behafteter und daher unsicherer Systeme eingebaut werden. Machine-Learning-Komponenten komplizieren dies weiter. Die Verstehbarkeit von Software, also insbesondere die Nachvollziehbarkeit der Entscheidungen und Kontrolle, die Software trifft bzw. ausübt, wird daher in der nächsten Zeit deutlich wichtiger. Dabei ist Nachvollziehbarkeit während der Nutzung aber auch „post mortem“, zum Beispiel zur Verbesserung der Software oder für gerichtliche Klärungen, notwendig. Sollte das nicht nachhaltig gelingen, werden roboterpsychologische Analysen für das Verhalten intelligenter Maschinen im Sinne von Asimovs notwendig.

Während die anderen Ingenieurswissenschaften im Wesentlichen die Naturwissenschaften und die Mathematik als Grundlagenwissenschaften nutzen, sind dies für das Software Engineering vor allem die Mathematik, aber auch die Sozialwissenschaften, weshalb eine Befruchtung in beide Richtungen (zwischen Software Engineering und Sozialwissenschaften) sinnvoll und notwendig ist.

Was macht moderne Software-Engineering-Forschung aus und welche Themen müssen diskutiert werden?

Um aktuelle und zukünftige Forschungsbeiträge des Software Engineering abzustecken und den Bezug zu anderen Wissenschaften zu präzisieren, ist es notwendig, die moderne Software-Engineering-Forschung im Zeitalter der Digitalisierung zu charakterisieren.

Abb. 1 zeigt die Kernbereiche der Software-Engineering-Forschung und ihre Wechselbeziehungen zu Anwendungsdomänen und zu Grundlagenwissenschaften. Im Sinne der oben genannten Kontextorientierung des Software Engineering [2] und der Durchdringung aller Lebensbereiche mit Software liefern Anwendungsdomänen wie die Produktion, die Logistik, das Gesundheitswesen, aber auch Wissenschaftsdisziplinen selbst (insbesondere auch die Informatik) den Kontext, d. h. Daten und Problemstellungen aus einem bestimmten Sachzusammenhang, für die Software-Engineering-Forschung. In der Software-Engineering-Forschung werden Artefakte (etwa Werkzeuge oder Entwicklungsmethoden) und Evidenz, d. h. empirisch oder formal fundierte Einsichten, bereitgestellt, um die Problemstellung im Anwendungskontext zufriedenstellend zu adressieren.

Als Ingenieursdisziplin greift die Software-Engineering-Forschung dabei neben eigenen Forschungsansätzen auf Grundlagenwissenschaften wie Mathematik, Informatik, andere Ingenieurswissenschaften oder Sozialwissenschaften zurück. Diese liefern Algorithmen und Theorien und unterstützen dabei das Design von Lösungen und die Wissensgewinnung im Software Engineering. In den folgenden Abschnitten behandeln wir den Kern des Software Engi-

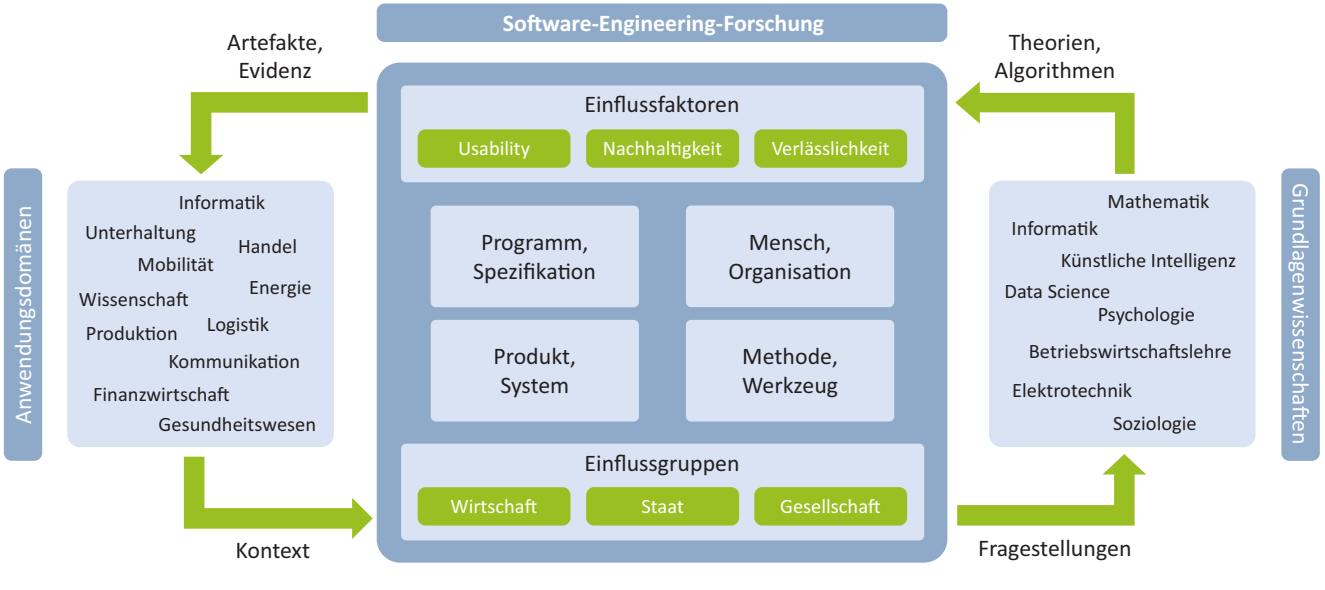


Abb. 1 Software-Engineering-Forschung sowie Verbindung zu Anwendungsdomänen und Grundlagenwissenschaften

neering, also auch den Bezug zu Anwendungsdomänen und Grundlagenwissenschaften ausführlich.

Kern der Software-Engineering Forschung

Software Engineering und die zugehörige Forschung sind seit langem klar definiert, daran ändern auch aktuelle Diskussionen wie hier oder im Zuge von 50 Jahre Software Engineering [1] nichts. Die folgende, bereits 1990 im IEEE Glossary of Software Engineering Terminology [5] vorgenommene Definition ist auch heute noch gültig und muss aus Sicht der Autoren nicht adaptiert werden:

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
2. The study of approaches as in (1).

Sie definiert das Software Engineering als Ingenieursdisziplin („application of engineering to software“) mit einer eigenen Methodik („systematic, disciplined, quantifiable approach“), angewendet auf alle Phasen des Lebenszyklus von Software („development, operation, and maintenance of software“). Der zweiteilige Aufbau der Definition bringt auch die enge Verzahnung von Software Engineering (1) und Software-Engineering-Forschung (2) zum Ausdruck. Die Grenzen zwischen beiden verschwimmen bisweilen. In der Software-Engineering-Forschung ist auf alle Fälle die systematische Reflexion deutlich ausgeprägter und der angestrebte Nutzen sowie die Generalisierbarkeit der Ergebnisse jenseits einzelner Projekte deutlich größer. Eine scharfe Trennung kann jedoch nicht gezogen werden, da zum Beispiel die Entwicklung eines Frameworks, Werkzeugs oder Meta-Modells gleichzeitig sowohl Anwendungs- als auch Forschungscharakter haben kann.

Klassischerweise differenziert sich Software-Engineering-Forschung selbst in die spezifische Betrachtung der unterschiedlichen Aktivitäten im Softwarelebenszyklus. Diese sind Analyse, Design, Implementierung, Testen, Deployment und Wartung. Hinzu kommen Querschnittsaktivitäten wie Projektmanagement, Qualitätsmanagement, Modellierung und Modellierungssprachen oder Variantenmanagement, die ebenfalls Teil der Software-Engineering-Forschung sind. Den zentralen Untersuchungsgegenständen in all diesen Kern- und Querschnittsaktivitäten des Softwarelebenszyklus können vier Bereichen zugeordnet werden:

1. Das Programm (der Code oder dessen Ausführung) und die Spezifikation (das informelle Diagramm, das formale textuelle oder grafische Modell, die natürlichsprachliche Anforderung)
2. Der Mensch (als Entwickler, Anwender oder in einer der vielen weiteren Rollen) und die Organisation (Entwicklungsteam, Integration in eine Gesamtorganisation)
3. Das Produkt (etwa die softwaregetriebene Innovation, die Qualität, die Softwarelogistik) oder System (etwa die Integration von Software mit Hardware oder die Integration von Systemen)
4. Die Methode (das Verfahren, die menschliche Aktivität, der automatisierte Bearbeitungsalgorithmus im Werkzeug, Prozessmodelle) oder das Werkzeug zur Unterstützung aller Phasen des Softwarelebenszyklus (etwa Modellierungs- oder Testwerkzeuge)

Als Ingenieursdisziplin folgt das Software Engineering sinnvollerweise einem Design-Science-Ansatz, bei dem Artefakte sich auf alle vorhin genannten Untersuchungsgegenstände des Software Engineering beziehen können. Der Begriff „Artefakt“ ist dabei relativ breit angelegt und schließt Werkzeuge, zum Beispiel automatisierende oder unterstützende Algorithmen etwa zur Testfallgenerierung, Templates zur Dokumentation von Anforderungen oder eine Methodik für die Aufwandsschätzung von Softwareprojekten mit ein. Mit der Durchdringung diverser Produkte und Dienstleistungen mit Software nimmt neben der systematischen Konstruktion von Artefakten die kreative Innovation eine immer bedeutendere Rolle ein und wird auch zunehmend durch Ansätze wie das Design Thinking oder Continuous Experimentation unterstützt. Werkzeuge spielen eine herausragende Rolle im Software Engineering, da sie die entwickelten Artefakte für die praktische Nutzung zugänglich machen. Erst die Existenz bestimmter Arten von Werkzeugen haben moderne Entwicklungsmethoden, wie zum Beispiel agile Entwicklung, möglich gemacht. Extreme Programming wäre ohne leichtgewichtige Testinfrastruktur, schnelle Generatoren und Übersetzer sowie Entwicklungsumgebungen mit integrierten Refactoring-Werkzeugen undenkbar. Der Komfort von spezifischen Editoren für Programmiersprachen (typspezifische Autovervollständigung, Highlighting, Refactoring im Editor) hat die Programmierweise der Entwickler massiv verändert. Hätten wir solche Werkzeuge auch in den früheren Phasen, wären auch Requirements Engineering, Architektur und Design wesentlich effizienter. Auch fehlen uns gute Werkzeuge zur Nachverfolgung von Anforderungen und Designentscheidungen, die man dringend benötigt, um den Impact von Änderungen zu antizipieren.

Software Engineering bzw. die Anwendung in konkreten Projekten beinhaltet typischerweise eine ausgeprägte Reflexion, die sowohl zur Optimierung innerhalb des Projekts als auch zur Weitergabe von projektübergreifendem Wissen genutzt werden. Daraus ergibt sich die Konsequenz, dass ein Teil der Software-Engineering-Forschung notwendigerweise im industriellen Kontext vorgenommen werden muss

und es deshalb des Einsatzes empirischer Methoden bedarf, um im industriellen Kontext Evidenz abzuleiten, etwa um Entscheidungen zu unterstützen, welches Prozessmodell in welcher Adaptierung in welchem Kontext am effektivsten funktioniert. Das „Manifest für Agile Softwareentwicklung“ [7] etwa wurde 2001 auf einem Treffen von Praktikern formuliert und von diesen in die Praxis der Softwareentwicklung getragen. Die Aufgabe der Software-Engineering-Forschung ist es, in diesem Zusammenhang die Methoden wissenschaftlich fundiert weiterzuentwickeln und deren praktischen Einsatz kritisch und empirisch fundiert zu hinterfragen und zu bewerten, um sie dadurch abhängig vom Anwendungskontext durch die Bereitstellung von Evidenz zu optimieren.

Aufgrund der Anwendungsorientierung und der durch Software getriebenen Digitalisierung spielen auch Einflussfaktoren und Einflussgruppen für die Software-Engineering-Forschung eine große Rolle. Zu den Einflussfaktoren gehören die Verlässlichkeit („Dependability“), die Nachhaltigkeit („Sustainability“) und die Benutzerfreundlichkeit („Usability“). Verlässlichkeit ist traditionell stark vom Einsatz softwarebasierter Systeme in kritischen Domänen wie der Luftfahrt beeinflusst und umfasst Attribute wie Zuverlässigkeit („Reliability“), Sicherheit („Safety“), Verfügbarkeit („Availability“ und damit letztlich auch Performanz), Wartbarkeit („Maintainability“) und Informationssicherheit („Security“), aber zunehmend auch Nachvollziehbarkeit und Erklärbarkeit, speziell im Kontext moderner datengetriebener Anwendungen. Ausgehend vom ökologisch geprägten schonenden Umgang mit Ressourcen beinhaltet die Nachhaltigkeit auch ethische Aspekte wie die Fairness von Entscheidungsalgorithmen. Usability steht für die Benutzerfreundlichkeit oder Gebrauchstauglichkeit eines softwarebasierten Produkts oder eine Dienstleistung und gewinnt insbesondere mit neuen Benutzerschnittstellen zunehmen an Bedeutung.

Als Einflussgruppen spielen die Wirtschaft, der Staat und die Gesellschaft eine große Rolle, etwa indem sie den rechtlichen Rahmen für die Entwicklung und den Einsatz von Software abstecken, wie zum Beispiel durch die Datenschutzgrundverordnung in Bezug auf die Verarbeitung personenbezogener Daten.

Relation des Software Engineering zu ihren Grundlagenwissenschaften

Als Ingenieurswissenschaft ist das Software Engineering wie andere Ingenieurswissenschaften auch auf die Erkenntnisse von Grundlagenwissenschaften angewiesen. Für die anderen Ingenieurswissenschaften sind im Wesentlichen die Naturwissenschaften und die Mathematik die Grundlagenwissenschaften, im Software Engineering ist die Situation durch den immateriellen und den immanent humanen Charakter komplexer. Natürlich sind die Informatik, die etwa genetische Algorithmen für das Search-Based Software Engineering oder die damit verwandte künstliche Intelligenz bereitstellt, und das Data Science als Datenwissenschaft wichtige Grundlagendisziplinen für das Software Engineering. Mit der zunehmenden Bedeutung der cyberphysikalischen Systeme sind auch andere technische Wissenschaften wie die Elektrotechnik oder der Maschinenbau bzw. die Physik als deren Grundlage wichtige Grundlagendisziplinen. Wie bereits erwähnt sind aber auch die Wirtschafts- und Sozialwissenschaften wie die Betriebswirtschaftslehre, die Soziologie oder die Psychologie wichtige Grundlagenwissenschaften für die Software-Engineering-Forschung. Wenn demnächst biologische Informationsverarbeitung stärker eine Rolle spielt, wie das heute Human-Brain-Projekte oder Modellierung biologischer Systeme mit Informatikmethoden andeuten, werden auch Biologie und Chemie als Grundlage noch relevanter.

Der enge Bezug des Software Engineering zu Grundlagenwissenschaften ist bereits im Software Engineering Body of Knowledge (SWEBoK) [6] sichtbar, in dem eigene Kapitel Computing Foundations, Mathematical Foundations und Engineering Foundations im Wesentlichen auf die Grundlagenwissenschaften Informatik, Mathematik und Ingenieurswissenschaften verweisen.

In der Software-Engineering-Forschung entstehen dabei bestimmte Fragestellungen, die mithilfe der Grundlagenwissenschaften adressiert werden. Diese stellen Algorithmen oder Theorien bereit, die formal-mathematischer, physikalischer oder sozialwissenschaftlicher Natur sein können, die im Forschungsprozess des Software Engineering in die Entwicklung und Evaluierung von Lösungen einfließen und die als Artefakte und Evidenz wieder in die Anwendungsdomänen zurückfließen können.

Je nach betrachtetem Untersuchungsgegenstand kann Software Engineering daher sehr mathematiknah und analytisch formal sein, zum Beispiel, um formale Modelle oder Programme zu behandeln, aber auch sehr empirisch und experimentell, um etwa spezifische Hypothesen über den menschlichen Faktor bei der Entwicklung und der Nutzung zu untersuchen. Die dritte große Säule des Erkenntnisgewinns, die Simulation, ist im Software Engineering als Test bereits intensiv im Einsatz und wird bei integrierter Qualitätssicherung von Software (beispielsweise durch Architektur- oder Verhaltenssimulation) in einem komplexen System wie etwa einer Smart City auch in anderen Formen noch intensiver eine Rolle spielen. Im Software Engineering fehlen aber immer noch Eigenschaften klassischer Ingenieursdisziplinen, wie der durchgängigen modellbasierten Simulation vor der Realisierung zur frühen Qualitätsbewertung. Abgesehen von einzelnen Ansätzen (wie Software-Architektursimulation) ist allerdings auch unklar, ob eine solche frühe Simulation auf Modellbasis zur Qualitätsbe-

wertung von den klassischen Ingenieursdisziplinen auf das Software Engineering direkt übertragbar ist, da ja Software selbst im Gegensatz zu physischen Artefakten wie Brücken einen immanenten Modellcharakter hat.

Software und damit auch das Software Engineering ist seit jeher eine Integrationsdisziplin. Dies gilt natürlich für Produkte ganz besonders, die typischerweise aus Subsystemen und Komponenten integriert werden. Dies gilt aber auch für Entwicklungsmethoden, in denen je nach Art des Subsystems unterschiedlichste Methoden anderer Informatikteildisziplinen zum Einsatz kommen. SoftwareingenieurInnen sind daher Generalisten. Demgegenüber gibt es praktisch für jeden Teilaspekt der Software jeweils eigenständige Spezialistenrollen. Dies ist so im Bereich der Netzwerke, der Betriebssysteme, der Datenbanken, der eingebetteten Systeme, der Cloud, des Übersetzerbaus, des Entwickelns von (domänenspezifischen) Modellierungssprachen, der Nutzerinteraktion und natürlich auch der Sprach- und Bilderkennung mit statistischen Methoden oder maschinellem Lernen. Software Engineering greift deshalb nicht nur auf Grundlagendisziplinen wie die Informatik zu, sondern erweitert und bereichert diese auch. Aus diesem Grund sind die Informatik im Speziellen und Wissenschaft im Allgemeinen auch als Anwendungsdomänen in Abb. 1 angeführt.

Relation des Software Engineering zu ihren Anwendungsdomänen

Die großen Problemstellungen und Herausforderungen der Softwareentwicklung und damit auch der Software-Engineering-Forschung sind heute stark von der Anwendungsdomäne beeinflusst. Der Begriff Anwendungsdomäne kann dabei relativ breit gefasst werden und umfasst klassische Domänen wie Finanzen, Gesundheitswesen, Mobilität wie Automotive oder Avionik, Logistik, Unterhaltung, Handel oder Produktion. Darüber hinaus kann aber auch jede wissenschaftliche Disziplin wie die Informatik oder jede Sozial-, Ingenieurs- oder Naturwissenschaft, in denen heute größere Softwaresysteme im Einsatz sind, als Anwendungsdomäne verstanden werden.

Die Anwendungsdomänen stellen den Kontext, d. h. Daten und Problemstellungen aus einem bestimmten Sachzusammenhang, für die Software-Engineering-Forschung bereit. In der Software-Engineering-Forschung werden Artefakte (etwa Werkzeuge oder Entwicklungsmethoden) und/oder Evidenz, d. h. empirisch oder formal fundierte Einsichten bereitgestellt, um die Problemstellung im Anwendungskontext zufriedenstellend zu adressieren.

Mittlerweile ist klar, dass zwar die Fragestellungen für die Softwareentwicklung in allen Domänen sehr verwandt sind, aber die Antworten häufig sehr unterschiedlich ausfallen müssen. Sichtbar wird dies unter anderem bereits an den sehr heterogenen Technologie-Stacks und den verwendeten Programmiersprachen. Dementsprechend gibt es für viele Domänen jeweils eigene Ausprägungen des Software Engineering und damit sinnvollerweise auch der Software-Engineering-Forschung. Domänenspezifisches Software Engineering bedeutet immer auch intensive Zusammenarbeit mit den Domänenexperten.

Wo geht die Reise hin?

Digitalisierung des Software Engineering und des Systems Engineeering

Die aktuell stattfindende Digitalisierung praktisch aller Domänen erfordert domänenspezifische, effektive Verfahren, Software schnell zu erstellen und wiederverwendbar weiterzuentwickeln. Zu viele Domänen sind heute noch getrieben von den dort üblichen Entwicklungsmethoden, die typischerweise in Konflikt zur Softwareentwicklungsmethodik stehen und daher in Projekten regelmäßig Friktionen produzieren. Dies gilt insbesondere für die klassischen Ingenieursdomänen wie Maschinenbau, Bauingenieurswesen oder Elektrotechnik. Solange die Maschine das dominante Element des Produkts ist, muss natürlich der Entwicklungsprozess entlang der Maschine und ihrer Komponenten definiert werden und daher typischerweise geometrische Dekomposition anbieten. Da aber immer mehr die Softwarefunktionalität ansteigt und die Software der Innovationstreiber wird, ist es sinnvoll, über eine Neuverhandlung der Entwicklungsprozesse nachzudenken, die eine funktionale Dekomposition entlang der Softwarefunktionalität stärker in den Fokus nehmen. Die heute üblichen Softwareentwicklungsmethoden, wie etwa das V-Modell oder agile Methoden, sind jedoch auch nicht direkt auf andere Disziplinen übertragbar, obwohl die beobachtbare und durchaus berechtigte Hoffnung besteht, dass eine Generalisierung von Software-Engineering-Methoden für andere Domänen funktionieren dürfte. Es ist eine herausragende Aufgabe des Software Engineering in Kooperation mit den traditionellen Ingenieursdisziplinen, integrierte, ganzheitliche Entwicklungsmethoden und darauf aufbauende Werkzeuge zu entwickeln. Allerdings ist die Entwicklung eines Werkzeugs, das von Praktikern zur Anwendung genommen wird, heute keineswegs trivial. Explizite Werkzeugforschung ist notwendig, um den Bau industriell nutzbarer Werkzeuge überhaupt zu ermöglichen, denn moderne und damit oft auch intelligente Werkzeuge könnten die Softwareentwicklung weit jenseits des heutigen Standes unterstützen. Dies gilt allerdings nur, wenn solche Werkzeuge einerseits eine gewisse Reife besitzen und andererseits durch explizit offene Schnittstellen und Modularisierungstechniken Erweiterbarkeit sowie projektspezifische Anpassbarkeit ermöglichen. Das Systems Engineering und speziell

der Einsatz von Modellierungs- und Simulationstechniken, speziell in der Qualitätssicherung, ist dabei eine wesentliche Herausforderungen, um die Zeit zur Marktreife sowie die Entwicklungskosten innovativ weiterentwickelter Produkte deutlich zu verkürzen. Dies ist gerade in Hochlohn-Ländern essenziell, um die Innovationsfähigkeit nicht zu verlieren.

Das Verhältnis des Software Engineering zu Daten

Natürlich erhalten auch Daten, die immer mehr domänenspezifische Information in sich tragen, in der Funktionalität von Software eine immer größere Rolle. Bisher wurden Daten zwar als Gegenstand der Softwarefunktionen verarbeitet, sie haben aber nicht selbst wesentlich die Funktionalität bestimmt, wie das zunehmend in modernen datenintensiven Systemen der Fall sein wird. Das ändert sich nun durch Verfahren des maschinellen Lernens. Dadurch entfällt die Möglichkeit, die Software zur Entwicklungszeit vollumfänglich zu analysieren oder zu testen. Deshalb sind Verfahren zur Absicherung selbstlernender, hoch adaptiver Software notwendig, die klären, unter welchen Bedingungen Systeme noch zum Entwurfszeitpunkt analysierbar sind, Verfahren zur Laufzeitabsicherung und Verfahren zur A-posteriori-Fehleranalyse. Darüber hinaus spielen Daten noch eine weitere Rolle als „first class entities“ bei der Modellierung von Privatheitsanforderungen und allgemeiner Zugriffsregelungen.

Die zunehmende Verfügbarkeit von Daten bietet auch für die Software-Engineering-Forschung eine große Chance, um Hypothesen empirisch zu prüfen. Allerdings birgt die beinahe beliebige Verfügbarkeit bestimmter Arten von Daten etwa aus Open Source Repositories und die sehr schwierige Erhebung anderer Datenarten etwa im Industrie 4.0-Kontext im Software Engineering auch Gefahren für den Forschungsprozess. Zu häufig werden Daten etwa aus Softwarerepositorien oder durch Umfragen generiert und publiziert, ohne diese durch eine Theorie, eine genaue Kontextdefinition oder eine Vision zu fundieren, wodurch der Nutzen dieser Arbeiten für die Disziplin des Software Engineering unklar bleibt. Mit Blick auf die Naturwissenschaften bringt der Philosoph Karl R. Popper mit seiner Aussage „Alle Erkenntnis ist theoriegetränkt, auch unsere Beobachtungen“ [8] zum Ausdruck, dass Experimente zur Erhebung von Daten gemäß einer zu prüfenden Hypothese aufgebaut sein sollen und diese Hypothese aus einer Theorie folgt. Eine Datenerhebung ohne zugrundeliegende Theorie mag zwar der initialen Phase eines Forschungsprozesses als explorative Studie durchaus berechtigt sein, um daraus Hypothesen abzuleiten. Diese sind dann aber durch weitere empirische Studien zu evaluieren. Dazu kommt, dass die Verfügbarkeit von vielen Open-Source-Repositorien den Blick darauf verstellt, dass Software nicht nur auf Code-Artefakten beruht.

Qualität: Verlässlichkeit, Nachvollziehbarkeit, Erklärbarkeit...

Verlässlichkeit (Dependability), also der Grad, in dem darauf vertraut werden kann, dass ein System zugesicherte Leistungen erbringt bzw. zugesicherte Eigenschaften hat, gewinnt in Bezug auf Zuverlässigkeit, Security und Safety von Systemen eine immer größere Bedeutung. Gleichzeitig wird es aber für stark vernetzte, komplexe und autonome Systeme immer schwieriger, Verlässlichkeit und darüber hinaus Nachvollziehbarkeit und Erklärbarkeit von Systemen (etwa auf zur gesellschaftlich notwendigen Klärung von Haftungsfragen bei schädigendem Verhalten autonomer Systeme, was als Nachhaltigkeitsaspekt verstanden werden kann) sicherzustellen. Vielversprechende Ansätze, um dieses Problem zu adressieren, sind Abstraktionstechniken basierend auf Modellen, die helfen, die Komplexität, Vernetztheit, Autonomie und Nachvollziehbarkeit beherrschbar zu machen. Modellbasierte Ansätze sollen dabei sowohl generierende Modelle, beispielsweise für das modellbasierte Testen, als auch prädiktive Modelle, basierend auf maschinellem Lernen, integrieren. Diese Abstraktionstechniken sollten es dann ermöglichen, etwa relevante Eigenschaften des Verhaltens von High-Frequency-Trading-Techniken in der Finanzwirtschaft abzubilden, um das Zusammenwirken von mehreren dieser Algorithmen auf einem offenen Markt mit zugehörigen Analysetechniken zu untersuchen. Eine Herausforderung ist hierbei, wie man zum einen von den konkreten Details der Algorithmen soweit abstrahieren kann, dass Geschäftsgeheimnisse gewahrt werden und die Analyse noch skaliert, aber zum anderen noch aussagekräftige Analyseergebnisse gewonnen werden können, die ein Mensch die Ergebnisse noch nachvollziehen kann. Das Beispiel High-Frequency-Trading aus der Finanzwirtschaft zeigt auch nochmals die Rolle der Domäne.

Anwendungsdomänen entwickeln immer eine Eigendynamik und erfordern jeweils eigene Ausprägungen des Software Engineering und damit sinnvollerweise auch der Software-Engineering-Forschung.

Konkrete Fragestellungen des Software Engineering

Aus den bisherigen Betrachtungen ergeben sich viele konkrete Fragestellungen, welche die Software-Engineering-Forschung adressieren soll, um das Software Engineering und deren Forschung voranzubringen. Beispiele für konkrete Fragestellungen sind etwa:

- Nachvollziehbarkeit und Erklärbarkeit von Software: Wie können auch bei nichtdeterministischen datenin-

tensiven Softwaresystemen Ergebnisse plausibel erklärt werden?

- Security als ganzheitliches Thema: Wie können Security-Aspekte geeignet im Softwarelebenszyklus berücksichtigt werden? Wie kann etwa eine Brücke aussehen zwischen abstrakten Sicherheitsmodellen mit formalen Nachweisen und realer Software, die in dynamischen und offenen Umgebungen evolviert sind?
- Software Engineering und Nachhaltigkeit: Welche Software-Engineering-Methoden werden benötigt, um Ressourcenmanagement zu steuern und verstehen zu können?
- Forschungsmethodik im Software Engineering: Was können empirische und formale Methoden im Forschungsprozess leisten? Welche Rolle nehmen Kontextfaktoren (insbesondere auch menschliche und organisatorische Faktoren) ein und welche Theorien benötigt das Software Engineering und wie sehen diese aus? Welche Instrumente können Design Science im Software Engineering am besten unterstützen?
- Automatisierung durch Werkzeuge: Das in der Softwareentwicklung noch sehr viel mehr automatisiert werden kann ist unstrittig. Dadurch steigen die Effizienz der EntwicklerInnen und die Qualität der Produkte. Wie kann das schlummernde Potenzial besserer und automatisierter Unterstützung gehoben werden? Welche Werkzeuge sind besonders hilfreich? Wie können die Werkzeuge gestaltet werden, damit die Einstiegshürde möglichst gering ausfällt? Wie können Forschungsprototypen zu kommerziell einsetzbaren Werkzeugen weiterentwickelt werden?
- Qualitätssicherung für datenintensive Systeme: Mit welchen Methoden kann die Qualität von Softwaresystemen, deren Verhalten durch Daten und die Umgebung bestimmt wird und das zur Entwicklungszeit noch nicht feststeht, geeignet sichergestellt werden? Wie können Simulation, modellbasierte Softwareentwicklung, Runtime-Monitoring sowie statische Analyse und dynamisches Testen dabei geeignet integriert werden?
- Software in Systems-of-Systems: Wie geht man in Entwicklungsprozessen damit um, dass es keine zentrale Organisation mehr für Anforderungen und Entwicklungskoordination gibt? Was bedeutet Konsistenz in solchen Systemen und wie kann man sie wiederherstellen? Wie bringt man unterschiedliche Lebenszyklusmodelle und Updategeschwindigkeiten in Einklang? Wie integriert man die lebenslange Verbindung (Messung, Update) zwischen dem produzierten und ausgelieferten Produkt sowie seinem Nutzer und Hersteller?
- Umgang mit immer komplexeren Entwicklungs- und Laufzeitumgebungen: Wie kann man bei steigender Funktionalität EntwicklerInnen das methodische Wissen an die Hand geben, um solche Elemente einzusetzen bzw. zu entwickeln und zu pflegen?

Herausforderungen des Software Engineering

Das Software Engineering ist keine Natur-, sondern eine Ingenieurswissenschaft mit starkem Fokus auf die vier Säulen aus Abb. 1:

1. Programm und Spezifikation,
2. Mensch und Organisation,
3. Produkt und System sowie
4. Methode und Werkzeug.

Diese Säulen heben das Software Engineering von den traditionellen Ingenieurswissenschaften ab. Gerade wegen der am Entwicklungsprozess beteiligten Menschen treten anstelle von naturwissenschaftlichen Theorien oft Hypothesen, die sich aus Methodenvorschlägen ergeben und insbesondere auch menschliche und organisatorische Kontextfaktoren berücksichtigen. Diese Hypothesen über die Eigenschaften einer neu vorgeschlagenen Methode müssen dann empirisch geprüft werden, etwa durch kontrollierte Experimente oder Fallstudien. Um eine neu vorgeschlagene Methode im Software Engineering sinnvoll anwendbar zu machen und der empirischen Überprüfung zu unterziehen, ist es meistens notwendig, diese ganz oder teilweise in Werkzeugen zu implementieren. Nur so kann der Automatisierungsgrad der Methode, der durch ein Werkzeug ermöglicht wird, in Fallstudien eingebracht und empirisch validiert werden. Werkzeugbau ist daher eine notwendige und wichtige die Forschung begleitende Tätigkeit. Dass so zumindest auch Prototypen für Werkzeuge in der Softwareentwicklung entstehen, ist für die Industrialisierung neuer Software-Engineering-Methoden zumindest hilfreich und müsste auch politisch mehr unterstützt werden.

Es ist eine zentrale Aufgabe des Software Engineering, neue Visionen zu entwickeln und diese in Methoden und Werkzeuge zu gießen und begleitend deren Nützlichkeit, Effektivität und Effizienz empirisch zu evaluieren. Nur so ist es möglich, die tragende Rolle des Software Engineering für Wirtschaft und Gesellschaft im Zeitalter der Digitalisierung zu halten und weiter auszubauen. Gemäß der Aussage des Automobilfabrikanten Henry Fords „Wenn ich meine Kunden gefragt hätte, was sie wünschen, hätten sie gesagt: schnellere Pferde“, ist es notwendig, Visionen in praxistauglichen Methoden und Werkzeuge umzusetzen.

Dabei ist die Methodenweiterentwicklung gerade im Zusammenhang mit dem beschriebenen Wandel der Rolle des Software Engineering und ihrer stärkeren Verzahnung mit anderen Disziplinen (insbesondere dem Systems Engineering) außerordentlich spannend. War bisher die Entwicklung von Software ein Teilschritt im übergeordneten Systems Engineering, so wandelt sich dieses Verhältnis bei der enorm zunehmende Rolle von Software bei fast jeder technischen Innovation. Diese zunehmend zentrale Rolle der Software wird sich aus Sicht der Autoren auch in den

Methoden zur Systementwicklung niederschlagen – diese werden softwarezentrierter. Gemäß dieser neuen Rolle von Software bei Innovationen wird das Produkt durch seine Software getrieben, die elektrischen oder mechanischen Komponenten sind entweder Commodity oder werden entwickelt entsprechend der Anforderungen aus der Software. Dies bietet eine einzigartige Chance für das Software Engineering, als wichtiger Treiber im Systems Engineering aufzutreten. Diese Chance kann man nur nutzen, wenn sich Software-Engineering-Forscher ihrer Rolle als Ingenieurswissenschaftler und Innovatoren bewusst sind, deren Forschung Visionen für neuen Methoden hervorbringt.

Danksagung Unser herzlichster Dank gilt den Kolleginnen und Kollegen der Softwaretechnik und angrenzender Fachgebiete, deren Einschätzungen und Kommentare unter anderem auf den letzten Tagungen des Fachbereichs hier eingeflossen sind. Wir danken insbesondere (in alphabetischer Reihenfolge) Sven Apel, Jürgen Börstler, Michael Goedicke, Lars Grunske, Willi Hasselbring, Udo Kelter, Stefan Leue, Florian Matthes, Barbara Paech, Ina Schäfer, Klaus Schmid, Matthias Tichy, Walter Tichy und Stefan Wagner.

Funding Open access funding provided by University of Innsbruck and Medical University of Innsbruck.

Literatur

1. Erdogmus H et al (2018) 50 years of software engineering. IEEE Softw 35(5):20–24. https://doi.org/10.1109/MS.2018.3571240
2. Booch G (2018) The history of software engineering. IEEE Softw 35(5):108–114. https://doi.org/10.1109/MS.2018.3571234
3. Basili V et al (2018) Software engineering research and industry. IEEE Softw 35(5):44–49. https://doi.org/10.1109/MS.2018.290110216
4. Broy M (2018) Yesterday, today, and tomorrow. IEEE Softw 35(5):38–43. https://doi.org/10.1109/MS.2018.290111138
5. IEEE (1990) IEEE standard glossary of software engineering terminology
6. Bourque P, Fairley R (2014) SWEBOK V3.0 guide to the software engineering body of knowledge
7. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J, Thomas D (2002) Agiles manifest. https://agilemanifesto.org/iso/de/manifesto.html. Zugegriffen: 3. November 2020
8. Popper KR (1974) Objektive Erkenntnis. Ein evolutionärer Entwurf, 2. Aufl.

Informatik Spektrum (2021) 44:95–103
https://doi.org/10.1007/s00287-021-01349-9

HAUPTBEITRAG

Informatikunterricht in Deutschland – eine Übersicht

Richard Schwarz[1] · Lutz Hellmig[1] · Steffen Friedrich[2]

Angenommen: 3. März 2021 / Online publiziert: 6. April 2021

Zusammenfassung
Nunmehr 10 Jahre nach der letzten Synopse zur Situation des Informatikunterrichts in Deutschland wird mit dieser Untersuchung ein aktuelles Bild der informatischen Bildung in den 16 Bundesländern gezeichnet. Dem allgemeinbildenden Charakter der informatischen Bildung entsprechend liegt der Fokus auf Angeboten für einen verbindlichen – durch die Existenz eines curricularen Rahmens abgesicherten – Informatikunterrichts für alle Schülerinnen und Schüler in allen Schularten weiterführender Schulen. Die Grundlage für die Erhebungen bildeten sowohl umfangreiche Analysen bildungspolitischer Dokumente, Stundentafeln und curricularer Vorgaben als auch eine Verifikation der erhobenen Daten durch Expertinnen und Experten in den jeweiligen Bundesländern. Neben einer vergleichbaren Darstellung für jedes Bundesland machen die Übersichten zum Sekundarbereich I und II über alle Bundesländer deutlich, welche Defizite in der Grundlagenbildung zur Informatik noch immer vorhanden sind und das Vorankommen in der Digitalisierung in Schulen zusätzlich hemmen.

Bedeutung des Unterrichtsfachs Informatik

Im Zeitalter der Digitalisierung verändert sich auch der Alltag in Schulen. Unterricht unterliegt einem Wandel und erfordert durch den Einsatz moderner digitaler Medien andere Rahmenbedingungen. Für viele Berufe ist der tägliche Umgang mit modernen Technologien bereits im eher geschlossenen Umfeld in den Schulen eine gute Orientierung. Diese Veränderungen sind Ausdruck eines Umbruchs in den Leitmedien, den wir schleichend bereits im Alltag erleben. Hier muss die Schule für die Allgemeinbildung Antworten finden. Und diese sind eben nicht nur auf das Vorhandensein digitaler Medien, deren Beurteilung und den Austausch von Materialien über verschiedene Plattformen beschränkt.

Aktuell werden verstärkt digitale Arbeitsweisen und damit verbundene IT-Kenntnisse gefordert, es wird vor einer digitalen Spaltung der Gesellschaft gewarnt und in sehr unterschiedlichen Facetten über Sinn und Unsinn von Bildung im Kontext von digitalen Medien und Informatik diskutiert. Spätestens seit der Veröffentlichung der ersten Ergebnisse vergleichender empirischer Studien wie TIMSS, PISA oder ICILS haben verschiedene Debatten und fast unzählige wissenschaftliche und journalistische Schriften dafür gesorgt, Interesse an einer sogenannten „digitalen Bildung“ zu wecken, ohne deren Gegenstand genauer zu beschreiben und damit wirkliche Antworten auf Fragen zu digitalen Phänomenen zu ermöglichen [1].

Die Fähigkeit zum bloßen Hantieren mit Arbeitsgeräten und Werkzeugen ist genauso wenig Zeichen einer Kompetenz im künstlerischen oder naturwissenschaftlichen Bereich, wie das Vermögen, Texte auf einer Schreibmaschine zu tippen, Zeichen eines kompetenten Sprachgebrauchs ist. Diese Metapher darf konsequenterweise auch auf die „digitale Bildung“ angewendet werden.

Auch hinsichtlich der informatischen Bildung ist nicht die Bedienung einer Anwendung oder eines Gerätes entscheidend, sondern es sind Konzepte zu hinterfragen, um Alltagsphänomene (oder schlicht: Erlebnisse bzw. notwendige Handlungen) zu verstehen und ihnen gestaltend begegnen zu können. Informatische Bildung und Nutzung digitaler Medien im Sinne einer Medienbildung befinden sich im Schulalltag in einem ständigen Begründungszwang, obwohl in Wirtschaft und Gesellschaft viele Anwendungen aus diesen Bereichen längst zur Normalität gehören. Gleichzeitig

Richard Schwarz
richard.schwarz@posteo.de

✉ Lutz Hellmig
lutz.hellmig@uni-rostock.de

Steffen Friedrich
steffen.friedrich@tu-dresden.de

[1] Universität Rostock, Rostock, Deutschland

[2] TU Dresden, Dresden, Deutschland

werden die öffentlichen Medien wie Presse und Fernsehen nicht müde, von Problemen in der Nutzung moderner Informationskanäle durch Kinder und Jugendliche zu berichten. Vielfältige Formen der Bildung in diesem Bereich wurden und werden gefordert. Computerführerschein, Medienpass und viele andere Wortschöpfungen reihen sich aneinander. Die Herausforderungen an die Bildung betreffen insbesondere die Interaktivität, die Kreativität oder auch das Gestalten. Dabei geht es nicht darum, „etwas mit digitalen Medien“ zu machen, sondern vielmehr darum, digitale Anwendungen selbst zu entwickeln und kreativ zu verändern. Dafür sind dann auch Modellieren und Realisieren (eigentlich Implementieren) erforderlich, Kernkompetenzen einer informatischen Bildung. Es erscheint in diesem Kontext überfällig, eine systematische Bildung – die Informatik und die digitalen Medien betreffend – einzufordern.

Es geht nicht um ein „Mehr an Informatikunterricht“, sondern um die Anerkennung für ein allgemeinbildendes Schulfach Informatik für alle Schülerinnen und Schüler überhaupt. Die bildungspolitischen Debatten um die Einführung von Fächern und die vorhandenen vollen Stundentafeln in allen Schularten sind eine Seite. Die andere Seite ist die aktuelle Notwendigkeit, essenzielle Grundlagen der Veränderungen in Wirtschaft und Gesellschaft, insbesondere die Ausprägung der dafür notwendigen Kompetenzen, didaktisch geeignet in der Allgemeinbildung für alle fest zu verankern.

> Schüler und Schülerinnen brauchen kein Spezialwissen, aber ein grundsätzliches Verständnis für Fragen der maschinellen Verarbeitung von Informationen und Daten. Sie sollten nicht nur eine „passive Konsumhaltung“ einnehmen, sondern beurteilen können, welche medizinischen, ökologischen und ethischen Folgen die Digitalisierung oder die künstliche Intelligenz haben. Sie sollten auch einschätzen können, welche Grenzen die Digitalisierung hat [2].

Da informatische Bildung sich nicht in der Bedienung einer Anwendung oder eines Gerätes erschöpft, sondern genau die notwendigen Informatikkonzepte hinterfragt, um Erlebnisse bzw. notwendige Handlungen mit digitalen Anwendungen – also Informatiksystemen – zu verstehen, muss sie im Kontext der Bildung in einer digitalen Welt unbedingt verortet werden.

> Es gibt in den Bundesländern große Unterschiede in der Einordnung eines Schulfaches Informatik in die jeweilige Stundentafel. Das betrifft neben den Sekundarbereich I in den weiterführenden Schulen vor allem Informatikkurse in der Oberstufe auf unterschiedlichem Niveau. Das Interesse und die in der Schule bereits vorliegenden Angebote sind in verschiedenen Studien der letzten Jahre bereits sichtbar geworden. Zur Verbesserung der informatischen Bildung sind Maßnahmen zur Realisierung eines durchgängigen Informatik-Unterrichts umzusetzen [3].

Ein Fach Informatik als Kern der Ausprägung digitaler Kompetenzen dient der Darstellung und Systematisierung von Begriffen und Grundzusammenhängen der Informatik sowie der Vervollständigung von Kenntnissen und Einsichten zu grundlegendem Allgemeinwissen. Der Informatikunterricht stellt (ähnlich dem Mathematikunterricht im Rahmen einer mathematischen Bildung) ein wichtiges systematisierendes Element dar. In den durch die Gesellschaft für Informatik e. V. publizierten „Grundsätzen und Standards für die Informatik in der Schule“ sowie den „Kompetenzen für informatische Bildung im Primarbereich“ [4–6] sind Kompetenzen dargestellt und erläutert, die Anforderungen an eine informatische Bildung beschreiben, Antworten zu Alltagsphänomenen ermöglichen und an Beispielen zeigen, was Informatikbildung erreichen soll. Die Debatte um Schulfächer ist in diesem Zusammenhang ein zeitloses Thema:

> Die Schulfächer sind eine künstliche Konstruktion, kommen im Leben selbst nicht vor, was vielfach als Beweis für die Lebensfremdheit der Schule gilt. Sie sind die einzige Möglichkeit, die komplexe Wirklichkeit des Lebens denkend zu ordnen. ... Bei der Festlegung eines Kanons muß schließlich auch bedacht werden, daß jedes Schulfach sich hinsichtlich seiner Themen und Methoden deutlich von den anderen unterscheiden muß, damit es sich den Schülern als ein in sich vernünftig bearbeitbarer Aspekt der Wirklichkeit darstellen kann [7, S. 24, 26].

Nur ein Schulfach Informatik kann in diesem Sinne die notwendigen Sach-, Methoden- und Handlungskompetenzen ausbilden und Defizite im vorhandenen Fächerkanon beseitigen. Letztlich ist informatische Bildung allerdings nur in einem Gesamtkonzept zu verwirklichen, das Beiträge zur Medienerziehung ebenso einschließt wie den Unterricht in einem eigenständigen Fach. Gegen ein Schulfach Informatik für alle spricht sachlich nichts. In der Argumentation wird oft angeführt, dass Programmierkenntnisse nicht für alle nötig seien. Diese eingeschränkte Sicht auf die Gegenstände des Faches ist dabei häufig durch die eigene Schulerfahrung entstanden. Informatik in der Schule ist einfach mehr.

Die benutzten Anwendungen und Programmiersprachen sind immer Werkzeuge zur Vermittlung von Inhalten der Informatik, zum Erlernen der Arbeitsmethodik des Faches und zum Beurteilen des Einsatzes der jeweiligen Systeme. So sind beispielsweise die fundierte Einführung in Standardsoftware, Strukturierung von Informationen und Abläufen, Suchstrategien im Internet, Aufbau und Funktions-

weise des Internets oder auch die Beschreibung einfacher, automatisierbarer Vorgänge ein Beleg für die positive Wirkung der Informatik im Kanon der Unterrichtsfächer. Insbesondere geht es nicht nur um das Produzieren mit digitalen Werkzeugen, sondern auch um die digitalen Produkte selbst. Schließlich werden Modellierungstechniken benötigt, um zu verschiedenen Themen ein vertieftes Verständnis zu erzeugen.

In Deutschland existiert das Schulfach Informatik in der Oberstufe der Gymnasien seit ca. 40 Jahren und hat aus sehr unterschiedlichen Gründen bisher keine bundesweite Anerkennung gefunden. Verschiedene Versuche, die Gegenstände eines Faches Informatik integrativ in anderen Fächern oder Profilen einzubinden, können inzwischen bundesweit als gescheitert erklärt werden. Weder die Grundbildung in den 1980er-Jahren noch eine starke gesellschaftswissenschaftliche Ausrichtung Anfang der 1990er-Jahre konnten die Bildung zur Informatik in den Schulen stabilisieren. Die Chance, begleitend zum umfangreich geförderten Projekt „Schulen ans Netz“ die dazu gehörende Informatik-Bildung verpflichtend zu etablieren, wurde Mitte der 1990er-Jahre vertan. In der Vorstudie zur Bildungsinitiative „Schulen ans Netz“ wurde festgestellt:

> **Fünftes Ziel:** Sich-Aneignen von Hintergrundwissen Sich-Aneignen von Hintergrundwissen zum kompetenten und verantwortungsbewussten Umgang mit Netzen, d. h. das Erlangen bzw. Ergänzen einer informatischen Bildung [8, S. 25].

Betrachtet man den Zeitraum und eine Reihe gerade in jüngster Vergangenheit veröffentlichter Vergleichsstudien oder die internationalen Aktivitäten in diesem Bereich, wird erst richtig deutlich, welches Potenzial in den letzten 25 Jahren in Deutschland verschenkt wurde. Erst mit technischen und gesellschaftlichen Veränderungen Anfang der 2000er-Jahre wurde die Debatte bildungspolitisch wieder relevant, ohne bis heute einen Durchbruch zu einer verpflichtenden Informatikbildung zu erreichen.

Der aktuelle Stand zum Informatikunterricht ist in den Bundesländern sehr unterschiedlich und Vergleiche werden seitens der KMK nicht angestellt. Vorliegende Studien aus dem Jahr 2010 [9] sind wegen der positiven Veränderungen in jüngster Zeit nicht seriös zu verwenden. Es zeigt sich, dass man im Sekundarbereich in Mecklenburg-Vorpommern (seit 1998), Sachsen (seit 1992) und Bayern (seit 2001) ein Pflichtangebot in den einzelnen Jahrgangsstufen findet, in den beiden erstgenannten Ländern in allen Schulformen mit meist einer Wochenstunde. Mit großer Anstrengung aus den Schulen heraus hat sich die Situation in Baden-Württemberg (Pflicht in Klasse 7, Wahlpflichtangebot) etwas verbessert.

Das waren Gründe, um erneut eine Studie zum Stand des Informatikunterrichts in Deutschland an weiterführenden Schulen anzufertigen. Die Untersuchung erfolgte für die Sekundarstufe I (unter Einbeziehung der Orientierungsstufe ab Jahrgangsstufe 5) und für die Sekundarstufe II.

Untersuchungsdesign

Die Datengewinnung erfolgte durch eine umfangreiche, durch eine Expertenbefragung verifizierte Dokumentenanalyse.

Zunächst nahm die Sichtung curricularer Dokumente auf Inhalte mit Informatikbezug in den Bundesländern einen breiten Raum ein. Ein Curriculum trägt wesentlich zur Bestimmung des Stellenwerts eines Fachs bei. Durch ihren für Lehrkräfte bindenden Charakter haben curriculare Vorgaben in gewissem Maße Einfluss auf unterrichtliche Planungsprozesse und erlauben Rückschlüsse auf die Bildungspraxis [10, S. 12]. Konkret ergaben sich folgende Fragen.

- In welcher Form ist das dem curricularen Dokument zugrundeliegende Fach organisiert?
- Für welche Schulart bzw. welchen Schulbereich gilt das Dokument?
- Für welche Jahrgangsstufen gilt das Dokument?

Schon aus den curricularen Vorgaben ergaben sich Hinweise auf die Exklusivität und die Systematik der Vermittlung informatischer Bildung, die in die Kategorien „Eigenständiges Fach“, „Interdisziplinäres Fach“ sowie „Fächerübergreifender Unterricht“ gefasst werden können. Nur wenn Informatikunterricht im Rahmen eines eigenständigen Fachs stattfindet oder der Anteil informatischer Inhalte im Rahmen eines interdisziplinären Fachs explizit ausgewiesen ist, ist eine objektive Vergleichbarkeit der Stundenkontingente gegeben.

Vor diesem Hintergrund wurden für alle Bundesländer Recherchen zu bildungsbezogenen Dokumenten vorgenommen, die auf Gesetzes- oder Verordnungsebene die Verbindlichkeit, die Einbettung und den Umfang informatischer Bildung in den Schularten der Länder regeln.

Bei der Analyse der informatischen Bildung in der Sekundarstufe I lag besonderes Augenmerk auf verbindlichem Informatikunterricht für alle Schülerinnen und Schüler, unabhängig vom Bildungsgang. Im Einzelnen wurden in der Sekundarstufe I folgende Fragestellungen untersucht:

- Mit welcher Verbindlichkeit findet Informatikunterricht statt?
- In welchen Jahrgangsstufen findet Informatikunterricht statt?
- In welchem Stundenumfang findet Informatikunterricht statt?

Informatikunterricht in Deutschland (Sekundarstufe I) Stand: August 2020

Bundesland	Schulform → Abschluss	5	6	7	8	9	10	Prüfung	Hinweise und Anmerkungen
Baden-Württemberg	mittlere Reife	×	×	1	1	1	1		–
	mittlere und Hochschulreife	×	×	1	1	1	1		1)
	Hochschulreife	×	×	1	2	1	1		2)
Bayern	mittlere Reife	1	1	1	1	1	1		3)
	Hochschulreife	×	1	1	×	2	2		4)
Berlin	mittlere Reife	×	×	1	×	2	2		5)
	mittlere und Hochschulreife	×	×	1	×	2	2		
	Hochschulreife	×	×	1	×	2	2		
Brandenburg	mittlere Reife	×	×						6)
	mittlere und Hochschulreife	×	×						
	Hochschulreife	×	×			1	2		7)
Bremen	mittlere Reife	×	×	×	×	×	×		8)
	mittlere und Hochschulreife	×	×	×	×	×	×		
	Hochschulreife	×	×	×	×	×	×		
Hamburg	mittlere Reife	×	×						9)
	mittlere und Hochschulreife	×	×						
	Hochschulreife	×	×						10)
Hessen	mittlere Reife	×	×	×	×	×	×		8)
	mittlere und Hochschulreife	×	×	×	×	×	×		
	Hochschulreife	×	×	×	×	×	×		
Mecklenburg-Vorpommern	mittlere Reife	1	1	1	1	1	1	✓	11)
	mittlere und Hochschulreife	1	1	1	1	1	1	✓	
	Hochschulreife	1	1	1	1	1	1		
Niedersachsen	mittlere Reife	×	2	2	2	2	2		12)
	mittlere und Hochschulreife	×	2	2	2	2	2		
	Hochschulreife	×	2	2	2	2	2		13)
Nordrhein-Westfalen	mittlere Reife	×	×						14)
	mittlere und Hochschulreife	×	×						
	Hochschulreife	×	×	×	×				15)

Abb. 1 Übersicht zum Informatikunterricht in Sekundarstufe I (Stand August 2020). Die Zahlen in den Feldern entsprechen den Unterrichtsstunden pro Woche. 1) Gesamtkontingent 8 UStd./Wo. für Profilfach „Informatik, Mathematik, Physik“; Anteil Informatik insgesamt bei 3 UStd./Wo., 2) Analog mit Anteil Informatik bei 4 UStd./Wo., 3) Wahl- oder Wahlpflicht mit 2 UStd./Wo. in 9/10 möglich, 4) Fach „Natur und Technik“ in 6/7 mit verbindlichem Anteil Informatik; „Informatik“ in 9/10 nur an naturwissenschaftlich-technologischem Zweig, 5) In Jahrgangsstufe 7 oder 8 als ITG mit verbindlichen Anteilen; Einordnung in Pflichtbereich möglich; alternativ auch 3 UStd. in 9/10, 6) Schwerpunktunterricht über alle Jahrgangsstufen von 7–10 mit max. 5 UStd. möglich, meist nur in einer Jahrgangsstufe, 7) Schwerpunktunterricht über mehrere Jahrgangsstufen, meist 9/10 mit 1 bzw. 2 UStd./Wo., 8) In allen Schularten kein Informatikunterricht, auch nicht fächerübergreifend mit festen Anteilen, 9) Wenn angeboten, dann meist 2 UStd./Wo. insgesamt über alle Jahrgangsstufen, 10) Wie oben, Umfang nach Profil unterschiedlich, 11) Als Unterrichtsfach „Informatik und Medienbildung“; Wahlpflicht zusätzlich möglich, 12) Schwerpunktangebot auch nur in Jahrgangsstufe 9/10 möglich, 13) Profil mit Schwerpunktfach Informatik möglich, dann 3–4 UStd., 14) Gesamtkontingent von bis zu 14 UStd. über mehrere Jahrgangsstufen ab 2021: Pflichtfach Informatik in den Jahrgangsstufen 5/6, 15) Gesamtkontingent von 6 UStd. über 2 Jahrgangsstufen

In der gymnasialen Oberstufe haben die Bundesländer aufgrund der KMK-Vorgaben einen geringeren Handlungsspielraum als in der Sekundarstufe I. Vor dem Hintergrund der umfangreichen Wahlmöglichkeiten für alle Fächer und der besonderen Bedeutung der Prüfung in der gymnasialen Oberstufe sind die Fragestellungen für die Sekundarstufe II angepasst worden:

- In welcher Hinsicht werden Festlegungen zur Belegung des Fachs Informatik getroffen?
- Welche Bestimmungen zu den Abiturprüfungen gelten für das Fach Informatik?
- Ist das Fach Informatik den naturwissenschaftlichen Fächern gleichgestellt?

In der Sekundarstufe II erfolgte die Einteilung nicht nach Jahrgängen, sondern nach Orientierungsphase und Qualifikationsphase differenziert. Das erlaubt eine bessere Vergleichbarkeit zwischen den Ländern mit G8- und G9-Bildungsgang, führt zwangsläufig aber dazu, dass die Jahrgangsstufe 10 in Ländern mit G8 sowohl in der Übersicht

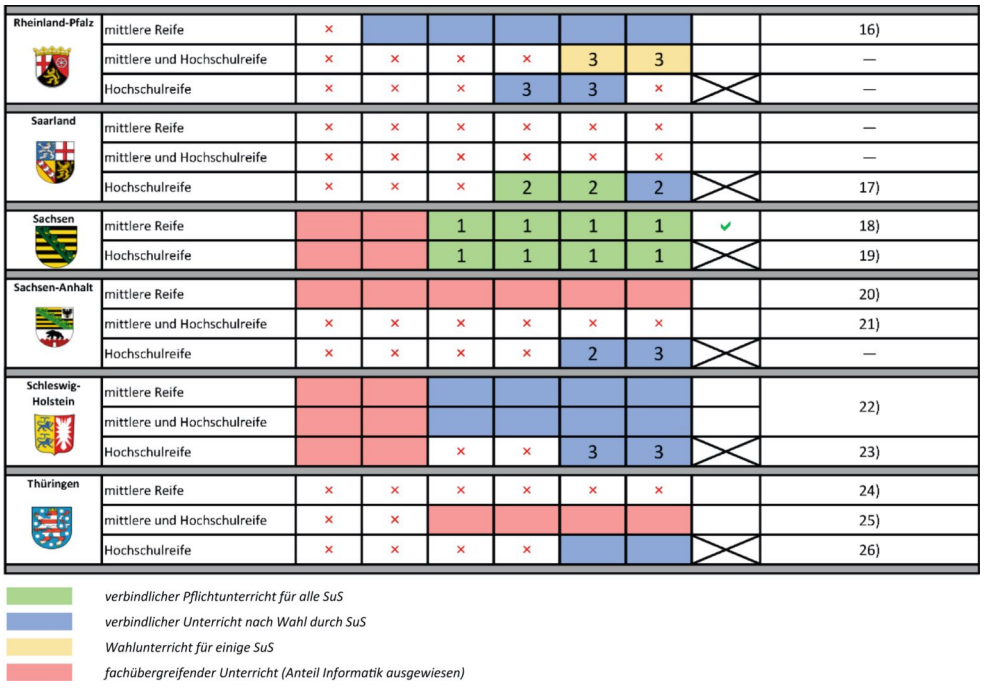


Abb. 1 (Fortsetzung) 16) Gesamtkontingent bis 6 UStd. als schuleigenes Angebot, 17) Gilt für MINT-Zweig (Schulversuch); 2 UStd./Wo. Pflicht in Jahrgangsstufe 8/9 bzw. alternativ 9/10, 18) In Jahrgangsstufe 5/6 „Technik/Computer" 0,4 UStd./Wo.; [M.I.T-Schulen mit zusätzlich 2 UStd./Wo. ab Jahrgangsstufe 8], 19) In Jahrgangsstufe 5/6 „Technik/Computer" 0,3 UStd./Wo.; [M.I.T.-Schulen: zusätzlich 3 UStd./Wo. ab Jahrgangsstufe 8], 20) Fach Technik im Profilbereich, Informatikanteile nicht verbindlich, 21) Im Gymnasialzweig analog zum Gymnasium möglich, 22) Gesamtkontingent von bis zu 16 UStd. über 4 Jahrgangsstufen für das Fach „Angewandte Informatik", 23) Gesamtkontingent von 6 UStd. über 2 Jahrgangsstufen für das Fach „Angewandte Informatik", 24) Fach Medienkunde über alle Jahrgangsstufen, keine verbindlichen Informatikanteile, 25) Gesamtkontingent von 8 bzw. 9 Ustd./Wo., als Fach „Wirtschaft-Recht-Technik"; Informatik anteilig (unverbindlich), 26) Gesamtkontingent von 6 UStd.

zur Sekundarstufe I als auch in der Übersicht zur Sekundarstufe II aufgeführt wird.

Im Ergebnis der Dokumentenanalyse wurde für jedes Bundesland ein vorläufiges Dossier zur informatischen Bildung angefertigt. Um Fehlinterpretationen zu vermeiden, wurden diese jeweils 2–3 Experten in den Bundesländern zugesandt, die die vorläufigen Dossiers verifizierten, ergänzten und weitere Hinweise zur inhaltlich und formal korrekten Darstellung der Situation gaben. Die verifizierten Dossiers sind bereits veröffentlicht worden [11, 12] und bilden die Grundlage für die aggregierte Darstellung der Situation der informatischen Bildung in Deutschland.

Ergebnisse der Datenanalyse

Eine Zusammenfassung der **Situation in der Sekundarstufe I** ist in Abb. 1 zu finden.

Hieraus ergeben sich folgende Kernaussagen zur Sekundarstufe I

1. In 2 Bundesländern – in Bremen und Hessen – existiert keinerlei Angebot für informatische Bildung im Sekundarbereich I.
2. In 9 Bundesländern wird Informatik ausschließlich nur im Wahlpflicht- und Wahlbereich angeboten.
3. Lediglich in den Bundesländern Baden-Württemberg, Bayern, Mecklenburg-Vorpommern, Saarland und Sachsen gibt es verbindlichen Informatikunterricht in der Sekundarstufe I.
   Hier sind jedoch z. T. enorme Unterschiede in der Breite des Angebots bezogen auf die Jahrgänge und betroffenen Schulformen auszumachen (Tab. 1).
4. Mecklenburg-Vorpommern ist seit 2019 das einzige Bundesland, in dem verbindlicher Informatikunterricht für alle Schülerinnen und Schüler durchgängig in den Jahrgangsstufen 5 bis 10 stattfindet.

Die Stundenkontingente für eigenständigen, verbindlichen Informatikunterricht können im Einzelfall eine große Bandbreite aufweisen, sodass verallgemeinernde Aussagen nur mit Einschränkungen zu treffen sind. Während es beispielsweise in Mecklenburg-Vorpommern und Sachsen eigenständigen, verbindlichen Unterricht an allen Schularten im Umfang von einer Wochenstunde gibt, findet dieser im Saarland nur in bestimmten Zweigen am Gymnasium statt; dafür kann dort der Informatikunterricht an Gymnasien im

Tab. 1 Bundesländer mit verbindlichem Informatikunterricht bezogen auf Zahl der Jahrgangsstufen und Schularten

	Verbindlicher Informatikunterricht in Jahrgangsstufen der Sek I		
	In allen	In mehreren	In einer
In allen Schularten	Mecklenburg-Vorpommern	Sachsen	Bayern, Baden-Württemberg
Nicht in allen Schularten, aber mindestens in einer	–	Saarland	–

Informatikzweig einen Gesamtumfang von 11 Wochenstunden aufweisen.

Ein Überblick zur **Situation in der gymnasialen Oberstufe** findet sich in Abb. 2.

Aus den Daten lassen sich folgende Aussagen zur **Belegung in der Einführungsphase** ableiten:

1. In 13 Bundesländern gibt es in der Einführungsphase Angebote für Informatikunterricht im Wahlpflichtbereich, wobei der Umfang zwischen 1 Wochenstunde und 4 Wochenstunden variiert.
2. In 3 Bundesländern gibt es in der Einführungsphase verbindlichen Informatikunterricht: Bayern (2 Wochenstunden), Mecklenburg-Vorpommern und Sachsen (je 1 Wochenstunde).

Zur **Belegung in der Qualifikationsphase**:

1. In allen Bundesländern kann Informatik in der Qualifikationsphase auf grundlegendem Anforderungsniveau im Umfang von 2 oder 3 Wochenstunden als Wahlpflichtmöglichkeit belegt werden. Eine Besonderheit stellt Baden-Württemberg dar: Dort ist ausschließlich eine zusätzliche Belegung im Umfang von 2 Wochenstunden möglich.
2. In 12 Bundesländern kann Informatik in der Qualifikationsphase auf erhöhtem Anforderungsniveau im Umfang von 4 oder 5 Wochenstunden angeboten und belegt werden.
   In Bayern, Baden-Württemberg, Sachsen und Sachsen-Anhalt kann Informatik in der Qualifikationsphase nicht auf erhöhtem Anforderungsniveau belegt werden.
3. In Niedersachsen, Sachsen und Thüringen wird Informatik bezüglich der Belegungsverpflichtungen in der Qualifikationsphase gegenüber den naturwissenschaftlichen Fächern gestärkt, da es im Rahmen länderspezifischer Bestimmungen explizit ein naturwissenschaftliches Fach ersetzen kann.

Bezüglich der **Abiturprüfungen** gilt:

1. In allen Bundesländern kann im Fach Informatik eine mündliche Abiturprüfung auf grundlegendem Anforderungsniveau abgelegt werden.
   (Mündliche Prüfungen auf erhöhtem Niveau sind lt. KMK-Vorgaben nicht möglich.)
2. Schriftliche Abiturprüfungen auf grundlegendem Niveau sind in 9 Bundesländern möglich.
3. Schriftliche Abiturprüfungen auf erhöhtem Niveau sind in 12 Bundesländern möglich.
4. Andere Prüfungsleistungen, wie z. B. Präsentationsprüfungen oder besondere Lernleistungen, sind in den Bundesländern Berlin, Brandenburg, Hamburg, Hessen, Mecklenburg-Vorpommern, Niedersachsen, Rheinland-Pfalz, Sachsen und Schleswig-Holstein möglich.

Fazit

Zum Informatikunterricht in Deutschland ergibt sich aus den Übersichten, vor allem für den Sekundarbereich I, ein sehr differenziertes Bild. Es zeigt deutliche Unterschiede im Hinblick auf die Verbindlichkeit, die jeweilige Schulart und vor allem die Organisationsform eines Fachunterrichts Informatik. Neben einem durchgängigen Pflichtunterricht in wenigen Bundesländern, für den die Inhalte curricular definiert sind, existieren unterschiedlichste Angebote im Wahlpflicht- bzw. im Wahlbereich (nach curricularen Vorgaben oder auch in freier Wahl der Schulen) oder es kommt ein solcher Unterricht überhaupt nicht zustande. Das kann einzelne Schularten, gymnasiale Zweige bzw. Schwerpunkte oder alle Schularten betreffen oder der Informatikunterricht kann in einer, in mehreren oder in allen Jahrgangsstufen stattfinden. Informatische Inhalte können zudem als eigenständiges Fach, im Rahmen eines interdisziplinären Fachs oder in Form fächerübergreifenden Unterrichts vermittelt werden. Diese Auflistung macht die Schwierigkeiten des Vergleichs deutlich, weil die Differenzen in der Struktur der Schullandschaft und in den Verantwortlichkeiten für Inhalt und Umfang zwischen den Bundesländern kaum vergleichbar darstellbar sind. Mit der Kategorisierung von Schulformen und der Art der hier gewählten Darstellung (einschließlich der Hinweise zu speziellen Regelungen) scheint ein gewisser Vergleich möglich, auch wenn dabei von einzelnen Facetten in den Bundesländern abstrahiert werden musste.

Bezugnehmend auf die letzte vergleichende Analyse [9] fällt das Ergebnis insgesamt ernüchternd aus. Die Forderungen von Wirtschaft und Gesellschaft, die die Gesellschaft für Informatik mit der Vorlage von Standards für den Informatikunterricht [4] bereits inhaltlich konkret untersetzt hat,

Bundesland	Anforderungsniveau	Orientierungsphase	Qualifikationsphase		Prüfung schr	Prüfung mdl	Gleich mit NatWi Belegung	Gleich mit NatWi Prüfung	Hinweise und Anmerkungen
Baden-Württemberg	grundlegend	1	2	2	×	✓	nein	nein	1)
	erhöht		×	×	×		×		
Bayern	grundlegend	2	2	2	✓	✓	nein	ja	2)
	erhöht		×	×	×		×		3)
Berlin	grundlegend	2	3	3	✓	✓	nein	ja	–
	erhöht		5	5	✓		nein		–
Brandenburg	grundlegend	2	3	3	✓	✓	nein	ja	–
	erhöht		5	5	✓		nein		–
Bremen	grundlegend	2	3	3	×	✓	nein	nein	4)
	erhöht		5	5	✓		nein		
Hamburg	grundlegend	2	2	2	✓	✓	nein	ja	5)
	erhöht		4	4	✓		nein		6)
Hessen	grundlegend	2	2	2	✓	✓	nein	ja	7)
	erhöht		5	5	✓		nein		
Mecklenburg-Vorpommern	grundlegend	1	3	3	✓	✓	nein	ja	8)
	erhöht		5	5	✓		nein		
Niedersachsen	grundlegend	3	3	3	✓	✓	ja	ja	9)
	erhöht		5	5	✓		ja		–
Nordrhein-Westfalen	grundlegend	2	3	3	✓	✓	nein	nein	–
	erhöht		5	5	✓		nein		–
Rheinland-Pfalz	grundlegend	2	3	3	×	✓	nein	ja	10)
	erhöht	4	5	5	✓		nein		–
Saarland	grundlegend	2	2	2	✓	✓	nein	ja	–
	erhöht		5	5	✓		nein		–
Sachsen	grundlegend	1	2	2	×	✓	ja	nein	–
	erhöht		×	×	×		×		11)
Sachsen-Anhalt	grundlegend	3	2	2	×	✓	nein	nein	–
	erhöht		×	×	×		×		–
Schleswig-Holstein	grundlegend	3	3	3	×	✓	nein	nein	–
	erhöht		4	4	✓		nein		12)
Thüringen	grundlegend	2	3	3	×	✓	ja	ja	–
	erhöht		5	5	✓		ja		13)

verbindlicher Pflichtunterricht für alle SuS
verbindlicher Unterricht nach Wahl durch SuS
Wahlunterricht für einige SuS
fachübergreifender Unterricht (Anteil Informatik ausgewiesen)

Abb. 2 Übersicht zum Informatikunterricht in Sekundarstufe II (Stand August 2020). Die Zahlen in den Feldern entsprechen den Unterrichtsstunden pro Woche. 1) Profilfach „Informatik, Mathematik, Physik“ mit 4 UStd./Wo. in der Orientierungsphase; Anteil Informatik ca. 1 UStd./Wo., 2) Gleichstellung in Prüfung nur am Naturwiss-Techn. bzw. WirtschaftsWiss. Gymnasium, 3) Wahlpflichtfach geplant, 4) Orientierungsphase auch 3 UStd./Wo., 5) Orientierungsphase (bei ausreichendem Kontingent); auch 4 UStd./Wo., 6) Nur als profilgebendes Fach, 7) Orientierungsphase ohne Zuordnung von UStd. (Festlegung durch Schule, 2–3 UStd./Wo.); Grundkurs auch mit 3 UStd./Wo. möglich, 8) Orientierungsphase: Fach „Informatik und Medienbildung“ in Jahrgangsstufe 10, 9) durchgängig nur im mathematisch-naturwissenschaftlichen Schwerpunkt, 10) Im Informatikzweig dreistündig als verpflichtendes Fach, 11) Ab 2023: M.I.T.-Schulen mit Leistungskurs (5 UStd./Wo.) und schriftlicher Abschlussprüfung, 12) Nur im naturwissenschaftlichen Profil wählbar, 13) Gilt nur für Spezialklassen an math.-naturwissenschaftlich-techn. Gymnasien

sind in den Bundesländern kaum realisiert. Ein Pflichtfach Informatik hat sich bundesweit – nach wie vor – nicht etabliert. Verbindlicher, eigenständiger Informatikunterricht ist sowohl regional als auch auf einzelne Jahrgangsstufen und Schularten beschränkt. Positiv hervorzuheben sind Mecklenburg-Vorpommern, das schulart- und jahrgangsübergreifend ein Pflichtfach „Informatik und Medien“ eingeführt hat, und Sachsen, wo ein eigenständiges Fach Informatik in den Jahrgangsstufen 7–10 für alle Schularten verbindlich ist. Schulartübergreifend gibt es in Baden-Württemberg ein Pflichtfach Informatik für eine Jahrgangsstufe. Dem allgemeinbildenden, weil alle Schularten betreffenden Anspruch, werden Bayern und das Saarland nicht gerecht. Hier gibt es vereinzelt an einigen Schularten bzw. in einigen Jahrgangsstufen verbindlichen Informatikunterricht. In allen anderen Bundesländern wird Informatikunterricht lediglich im Wahlpflicht- oder Wahlbereich angeboten. Bremen und Hessen sind negativ hervorzuheben, da im Sekundarbereich keine Angebote für Informatikunterricht existieren.

Anders ist die Situation, sicher auch historisch bedingt, für das Fach Informatik in der gymnasialen Oberstufe allgemeinbildender Schulen. Informatik kann in allen Bundesländern auf grundlegendem Anforderungsniveau belegt und als Prüfungsfach gewählt werden. In den meisten Fällen ist Informatik als Fach auf erhöhtem Anforderungsniveau wählbar. Im Rahmen einiger länderspezifischer Regelungen ist Informatik hinsichtlich der Belegungsverpflichtungen in der Qualifikationsphase der gymnasialen Oberstufe sowie als Abiturprüfungsfach den naturwissenschaftlichen Fächern gleichgestellt.

Diese Studie zum Informatikunterricht in Deutschland zeigt erneut, dass in den zurückliegenden Jahren viel über Digitalisierung und Medien diskutiert wurde, aber die Bedeutung einer informatischen Bildung für alle Schülerinnen und Schüler und deren Verbindlichkeit offensichtlich nicht erkannt wurde. Es fehlt eine breite Einsicht und Akzeptanz verantwortlicher Stellen sowie ein abgestimmtes und in der Schulpraxis realisierbares Konzept. Zu diesem Eindruck muss man bei einer Betrachtung der vielgestaltigen hexadezimalen Bildungslandschaft zur informatischen Bildung in Deutschlands Schulen kommen.

Bedenkt man, dass die Veränderungen mit der Industrialisierung vor über 100 Jahren eine Etablierung der Naturwissenschaften in den Schulen zur Folge hatten, könnte dies hinsichtlich des Handelns von heute eine Orientierung geben. Bei den Debatten um Medienbildung, um Datensicherheit und um mögliche Gefahren für Kinder und Jugendliche wird häufig übersehen, dass ein entscheidendes Defizit in der seit Jahren fehlenden Grundlagenbildung zur Informatik – also dem Schulfach Informatik – liegt und damit wichtige Kompetenzen zu den Konzepten der Digitalisierung fehlen. Erst in einem zeitgemäßen Informatikunterricht verstehen die Lernenden die Systeme in ihrer Lebenswelt. Er entmystifiziert Netzwerke, Datenbanken, Verschlüsselung und ebnet den Schülerinnen und Schülern den Weg zum mündigen Informationsbürger. Ein systematischer Fachunterricht legt außerdem die entscheidenden Grundlagen dafür, um in verschiedenen Kontexten konstruktiv die weitere Digitalisierung zu gestalten.

Gleichzeitig erscheint es anhand dieser Studie zur informatischen Bildung in Schulen in Deutschland und mit Blick auf die Entwicklung der Rahmenbedingungen erforderlich, einzelne Aspekte der Digitalisierung in Schulen bildungspolitisch zu diskutieren, um mittelfristig deutliche Veränderungen zu bewirken.

Mit diesen seit vielen Jahren existierenden Forderungen, gepaart mit konkreten Vorschlägen und dem Nachweis der Realisierbarkeit (in wenigen Bundesländern), stehen wir seitens der Gesellschaft für Informatik nicht allein. Zwei aktuelle Zitate, sowohl seitens der Wissenschaft in den Empfehlungen des Wissenschaftsrates zu den „Perspektiven der Informatik in Deutschland“ [13] als auch seitens der in dem Bereich aktiven gesellschaftlichen Kräfte im Rahmen der „Offensive Digitale Schultransformation“ [14], unterstreichen das erneut und zeigen, dass ein Handeln dringend erforderlich ist.

> **Der Wissenschaftsrat sieht informatische Bildung als zentralen Schlüssel** an, um den digitalen Wandel in der Gesellschaft erfolgreich, inklusiv und nachhaltig zu gestalten. ...
> Zwar beobachtet der Wissenschaftsrat einen recht weitgehenden Konsens in Gesellschaft, Politik und Wirtschaft darüber, dass eine Ausweitung der Vermittlung informatischen Wissens und informatischer Kompetenzen in der Schule vonnöten ist – konkret haben beispielsweise immer mehr Landesregierungen zuletzt beschlossen, in kleinen Schritten zukünftig verpflichtenden Informatik-Unterricht einzuführen. Zweifel sind angebracht, ob die Geschwindigkeit dieser positiven Entwicklung ausreicht angesichts der Tatsache, dass Deutschland noch weit entfernt ist von einer flächendeckenden und kontinuierlichen Vermittlung informatischer Bildung an alle Schülerinnen und Schüler. Auch im europäischen Vergleich besteht Nachholbedarf.
> Neben dem übergeordneten Ziel einer digitalen Mündigkeit aller Bürgerinnen und Bürger hält der Wissenschaftsrat schulische informatische Bildung aber auch vor dem Hintergrund der Notwendigkeit, mehr Absolventinnen und Absolventen von Informatik-Studiengängen hervorzubringen, für dringend erforderlich: Über Informatik-Unterricht in der Schule können bei Schülerinnen und Schülern frühzeitig Begeisterung für das Fach geweckt und ein diverserer Kreis an In-

teressierten für ein Studium gewonnen werden. Auch kann die schulische Bildung einen Beitrag zur Senkung der Studienabbruchsquoten im Studienbereich Informatik leisten, indem frühzeitig Grundkenntnisse und realistische Vorstellungen von einem Informatik-Studium vermittelt werden. Der Wissenschaftsrat ermutigt die Länder, die schnelle und flächendeckende Einführung informatischer Bildung in den Schulen noch stärker zu priorisieren, als dies bisher vorgesehen ist [13, S. 72].

Verpflichtenden Informatikunterricht ausweiten und die Nutzung von digitalen Werkzeugen in allen Fächern verbessern

Alle Kinder und Jugendliche, die auf IT-Systeme im schulischen und außerschulischen Kontext angewiesen sind, benötigen informatische Grundlagen, um anwendungsbezogene, technische und gesellschaftliche Perspektiven digitaler Technologien einschätzen zu können. Dazu sollte bundesweit flächendeckender Informatikunterricht ab der Sekundarstufe I angeboten werden, der auf alle drei Perspektiven eingeht und so die Medienbildung ergänzt und unterstützt [14].

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

1. KMK (2016) Kultusministerkonferenz: „Bildung in der digitalen Welt“ Beschluss vom 8. Dez. 2016 in der Fassung vom 7. Dez. 2017. Verfügbar unter: https://www.kmk.org/themen/bildung-in-der-digitalen-welt/strategie-bildung-in-der-digitalen-welt.html. Zugegriffen: 25. März 2021
2. Hellmig L (2019) Digitalisierung an Schulen: Informatik für alle – Eine Analyse von Julia Bernewasser. Zeit-Online vom 9. Mai 2019. Verfügbar unter: https://www.zeit.de/gesellschaft/schule/2019-05/digitalisierung-schulen-informatik-unterricht-programmieren-digitalpakt. Zugegriffen: 25.März 2021
3. Friedrich S (2017) Bildung in der digitalen Welt – Anmerkungen zum Strategiepapier der KMK. LOG IN 187/188:10–17
4. Gesellschaft für Informatik (2008) Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I. Empfehlungen der Gesellschaft für Informatik e.V. Beilage zu LOG IN, 28(150/151). https://www.informatikstandards.de. Zugegriffen: 25. März 2021
5. Gesellschaft für Informatik (2016) Bildungsstandards Informatik für die Sekundarstufe II – Empfehlungen der Gesellschaft für Informatik e.V. Beilage zu LOG IN, 36(183/184). https://www.informatikstandards.de. Zugegriffen: 25. März 2021
6. Gesellschaft für Informatik (2019) Kompetenzen für informatische Bildung im Primarbereich – Empfehlungen der Gesellschaft für Informatik e.V. Beilage zu LOG IN 39.(191/192). https://www.informatikstandards.de. Zugegriffen: 25. März 2021
7. Giesecke H (1999) Vom Sinn der Bildung. Funkmanuskripte 7. http://www.hermann-giesecke.de/Funk7.pdf. Zugegriffen: 25. März 2021
8. Schulen an das Netz (1995) Bildungsinitiative Informatik und Telekommunikation der Gesellschaft für Informatik e. V., Bonn (Sonderdruck). https://gi.de/themen/beitrag/macht-denselben-fehler-nicht-ein-zweites-mal. Zugegriffen: 25. März 2021
9. Starruß I (2010) Informatikunterricht in Deutschland. Analyse der informatischen Bildung an allgemeinbildenden Schulen auf der Basis der im Jahr 2010 gültigen Lehrpläne. Dresden. https://docplayer.org/50806106-Synopse-zum-informatikunterricht-in-deutschland.html. Zugegriffen: 25. März 2021
10. Vollstädt W et al (1999) Lehrpläne im Schulalltag: Eine empirische Studie zur Akzeptanz und Wirkung von Lehrplänen in der Sekundarstufe I. VS, Wiesbaden
11. Schwarz R (2020) Informatikunterricht in Deutschland. Darstellung der Situation in den 16 Bundesländern im Jahr 2020. Rostock. https://pidi.informatik.uni-rostock.de/lehre/abschlussarbeiten/hausarbeiten/. Zugegriffen: 25. März 2021
12. Gesellschaft für Informatik (2021) Informatik-Monitor. https://informatik-monitor.de. Zugegriffen: 25. März 2021
13. Wissenschaftsrat (2020) Perspektiven der Informatik in Deutschland. https://www.wissenschaftsrat.de/download/2020/8675-20.pdf. Zugegriffen: 25. März 2021
14. Offensive Digitale Schultransformation (2020) GI Berlin. https://offensive-digitale-schultransformation.de/. Zugegriffen: 25. März 2021

Informatik Spektrum (2021) 44:104–114
https://doi.org/10.1007/s00287-021-01343-1

HAUPTBEITRAG

Machine intelligence today: applications, methodology, and technology

Selected results of the 1st online Dagstuhl workshop on applied machine intelligence

Bernhard G. Humm[1] · Hermann Bense[2] · Michael Fuchs[3] · Benjamin Gernhardt[4] · Matthias Hemmje[4] · Thomas Hoppe[5,6] · Lukas Kaupp[1] · Sebastian Lothary[4] · Kai-Uwe Schäfer[7] · Bernhard Thull[1] · Tobias Vogel[4] · Rigo Wenning[8]

Accepted: 20 January 2021 / Published online: 18 February 2021

Abstract
Machine intelligence, a.k.a. artificial intelligence (AI) is one of the most prominent and relevant technologies today. It is in everyday use in the form of AI applications and has a strong impact on society. This article presents selected results of the 2020 Dagstuhl workshop on applied machine intelligence. Selected AI applications in various domains, namely culture, education, and industrial manufacturing are presented. Current trends, best practices, and recommendations regarding AI methodology and technology are explained. The focus is on ontologies (knowledge-based AI) and machine learning.

✉ Bernhard G. Humm
bernhard.humm@h-da.de

Hermann Bense
hb@bense.com

Michael Fuchs
michael.fuchs@wb-fernstudium.de

Benjamin Gernhardt
benjamin.gernhardt@fernuni-hagen.de

Matthias Hemmje
matthias.hemmje@fernuni-hagen.de

Thomas Hoppe
thomas.hoppe@fokus.fraunhofer.de,
thomas.hoppe@htw-berlin.de

Lukas Kaupp
lukas.kaupp@h-da.de

Sebastian Lothary
sebastian.lothary@fernuni-hagen.de

Kai-Uwe Schäfer
kai-uwe.schaefer@upilio.com

Bernhard Thull
bernhard.thull@h-da.de

Tobias Vogel
tobias.vogel@fernuni-hagen.de

Rigo Wenning
rigo@w3.org

[1] Hochschule Darmstadt—University of Applied Sciences, Haardtring 100, 64295 Darmstadt, Germany

[2] bense.com GmbH, Schwarze-Brüder-Str. 1, 44137 Dortmund, Germany

[3] Wilhelm Büchner Hochschule, Hilpertstraße 31, 64295 Darmstadt, Germany

[4] FernUniversität Hagen, Universitätsstr. 1, 58097 Hagen, Germany

[5] Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany

[6] Hochschule für Technik und Wirtschaft Berlin, 12459 Wilhelminenhofstr. 75a, Germany

[7] Frankfurt am Main, Germany

[8] World Wide Web Consortium, 2004 Route des Lucioles, 06902 Sophia Antipolis, France

Introduction

Machine intelligence, a.k.a. *artificial intelligence (AI)*[1], is one of the most prominent and relevant technologies today. It is in everyday use in the form of AI applications and has a strong impact on society.

In 2014, we started a series of annual workshops at the Leibniz Zentrum für Informatik, Schloss Dagstuhl, Germany, initially focusing on corporate semantic web, and later widening the scope to applied machine intelligence (AMI). In all workshops, we focussed on the application of AI technologies in corporate and organizational contexts. A number of books [1–3] and journal articles [4–7] resulted from those workshops. The workshops are characterized by an intense spirit of interdisciplinarity, collaboration, and focus on practical results [8]. Due to the coronavirus pandemic, the 2020 workshop was, for the first time, held online—however, this made it no less intense. It consisted of two half-day workshops with short presentations and parallel barcamp sessions. This article presents selected results from the 2020 workshop.

This article is structured as follows: In the next section we present selected AI applications in various domains, namely culture, education, and industrial manufacturing. The following section focuses on AI methodology, namely aspects of machine learning and knowledge representation. We then discuss selected technological issues of AI before concluding this article.

Current applications of AI

The following overview of AI applications in the domains of culture, education, and industrial manufacturing is not meant to be exhaustive, but shall demonstrate the diversity of AI by examples.

AI for performing arts

The term *performing arts* refers to ephemeral forms of art in which artists use their voices, bodies, or inanimate objects to convey artistic expression[2]. This comprises opera, theatre, ballet, concerts, and many other types of performances. As part of our cultural heritage, it is important to preserve works of performing arts. Developing archives for performing arts involves particular challenges. Due to its ephemeral nature, it is not possible to archive the performance itself. Instead, archives of performing arts contain artefacts in which a performing art work manifests itself, e.g., photographs, videos of performances, newspaper reviews, or interviews with contemporary witnesses (oral history). For major works, this leads to a large amount of typically unstructured material, which must be made accessible by archivists and historians. Examples of projects that have dealt with the development of archives for performing arts include:

- Development of the digital Pina Bausch archive of the Pina Bausch Foundation, Wuppertal, Germany (2011–2020): Development of an archive to represent the work of the choreographer Pina Bausch based on standards of Linked Data and Semantic Web, as well as CIDOC/Conceptual Reference Model (CRM) [9]. See Fig. 1.
- Development of the archive of the free theatre (dt. Archiv des Freien Theaters, 2017): Development of an approach to archive the work of more than 3000 individual artists and small companies who have performed theatre in Germany since the 1970s. The approach was also based on semantic web standards [10].
- Project study on the implementation of a live archive for the David Earle Dance Theatre, Toronto, Canada (2018–2019): Use of an archive to create new performances based on archived data by mixing real performances with archived material.

Which AI lessons can be learned from the development of archives for performing arts? Semantic web standards have proven to be well suited for these tasks. Additionally, we identified the need to apply pattern recognition algorithms. Due to high demands on data quality and accuracy, retrieval and formalization of data is done manually. However, this cannot usually be done for all collected material, the archive of the free theatre being the most striking example. It is therefore necessary to identify promising candidates within a huge search space, i.e, material that has not yet been described, but whose cataloguing is likely worthwhile. Here, the application of automated pattern recognition is a promising approach.

In the Pina Bausch Archive, videos are analyzed by marking scenes within a recorded performance. With more than 8000 video tapes and sometimes more than 100 scenes within one performance, this leads to a huge amount of work when carried out manually. In performances, scenes are usually identified with the help of cues, i.e., certain movements of actors, or changes in lighting or music. Automatic cue detection on videos can be of help here.

The reuse of material, e.g., to create new performances, creates the need to synchronize activities in real time. This

[1] We prefer the term machine intelligence to artificial intelligence (AI) in order to avoid interpretations of AI being a form of intelligence equivalent to human intelligence. However, we will use both terms interchangeably.

[2] Wikipedia: Performing arts. In Wikipedia, Accessed 6/11/2020 from https://en.wikipedia.org/w/index.php?title=Performing_arts&oldid=984658242.

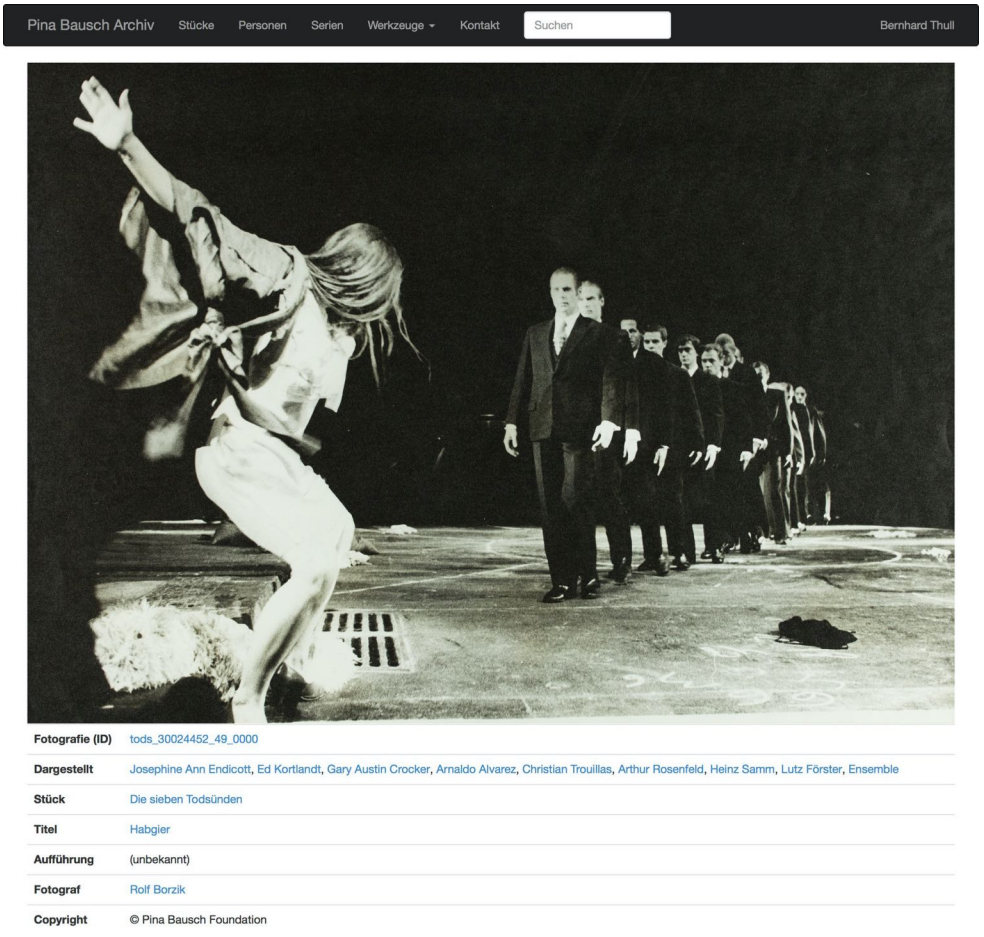


Fig. 1 Data browser for the digital archive of the Pina Bausch Foundation with a web page containing data on a photograph. By clicking on the corresponding links, users can find out more about the photograph itself, people depicted therein, the piece "The Seven Deadly Sins of the Petty Bourgeoisie," the title or the scene "Greed," and the photographer Rolf Borzik

is likewise done with the help of cues, and again, automatic cue detection on video and onstage is helpful.

Archiving material is a task lasting many years, carried out by institutions that are typically not equipped with IT departments. It is therefore a mandatory prerequisite that machine intelligence applications can be configured and used by lay people.

Building a semantic qualification web

The opportunities and offers for qualification and life-long learning are becoming ever more extensive. If you consider the academic sector alone, there are already more than 20,000 study courses offered by approximately 400 qualification institutes in Germany [11]. Consequently, planning individual qualifications may become a challenge. How might AI help in this scenario?

The vision of Tim Berners-Lee[3], the inventor of the WWW, was to create a data network in which software agents (SWA) can act [12]. SWAs are symbolic AI programs that can navigate the WWW and make autonomous decisions based on ontologies [13]. Examples of SWA tasks include booking trips, arranging medical appointments, or creating qualification paths, i.e., sequences of qualifying actions that lead to a certain qualification goal. This data network was named *semantic web (SeW)* and was introduced by Berners-Lee in the Scientific American 2002 [14, 15].

We use the term *semantic qualification web (SQW)* for the idea of a semantic network in the qualification sector. An SQW needs to model qualifications offered by organizations (e.g., universities, courses, and modules), as well as competences taught. Such information is usually provided in textual form on the websites of such organizations, e.g., in module descriptions. A representation in machine-readable form, e.g, as an ontology using SeW standards, is rare. Manually creating such an SQW ontology is extremely cost-intensive and constantly keeping it up-to-date is not feasible. Therefore, AI methods of natural language processing (NLP) using machine learning (ML) may be used to semi-automatically create an SQW ontology from texts provided on such websites.

Fig. 2 shows a simplified process for retrieving qualification offer texts from websites and transforming them into

[3] https://www.w3.org/People/Berners-Lee/ (accessed 4 Nov 2020).

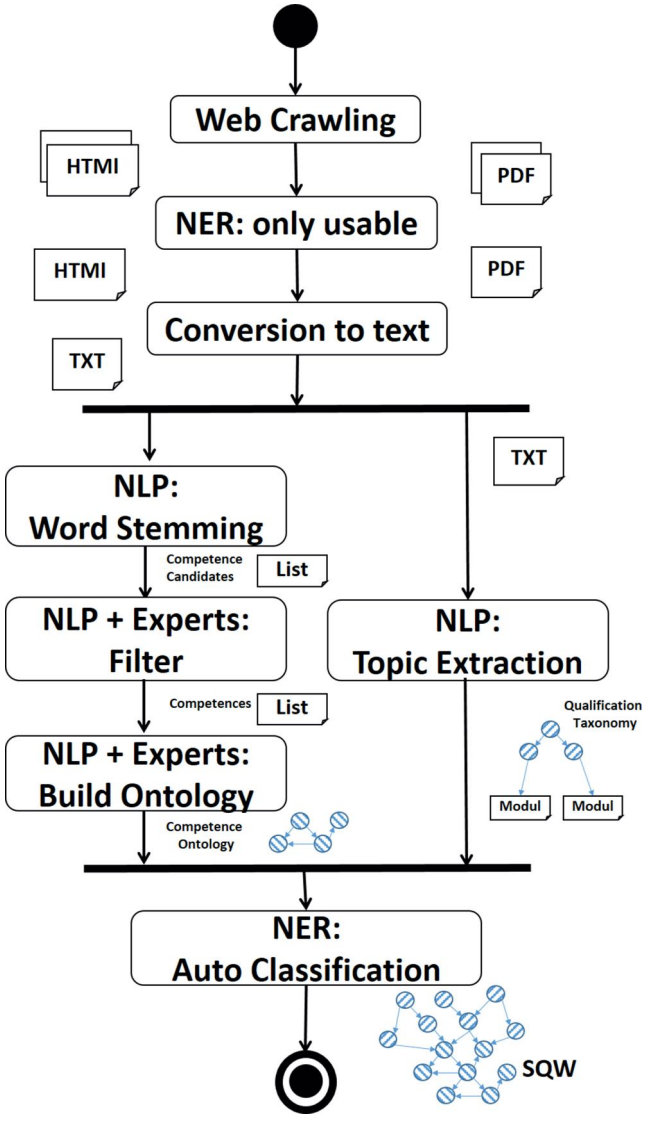


Fig. 2 Natural language processing (*NLP*)-based process for transforming text to semantic qualification web. *HTML* HyperText Markup Language document, *TXT* Text document, *PDF* Portable Document Format document, *NLP* Natural Language Processing, *NER* Named Entity Recognition, *SQW* Semantic Qualification Web

an SQW ontology. The process is shown as a UML Activity Diagram.

Web crawling can be used for systematically analyzing web pages, starting with yellow pages for qualification offers, e.g., Higher Education Compass[4]. Pages are automatically classified for relevance for the SQW, using the NLP technique of *named entity recognition (NER)* [16].

Other NLP approaches like Stemming[5] can be used for semi-automatically extracting a competence ontology. Domain experts need to manually support by filtering and building the competence ontology. In a parallel process, the NLP approach of Topic Extraction[6] may be used to build a qualification taxonomy.

Finally, NER-based auto-classification can be used for linking modules from the organizational taxonomy and the competencies from the competence ontology. Thus, the resulting SQW ontology connects the qualification taxonomy with the competence ontology.

This example shows how AI methods from NLP and ML can help create SeW ontologies.

Context-aware fault diagnosis in the smart factory

The *smart factory* forms a complex environment with highly coupled, integrated, and interconnected machinery. The detection of a fault is a complex task, especially in the transition between industry 3.0 with brownfield machinery and industry 4.0. In this transition, brownfield machinery exists side by side with new *cyber-physical systems (CPSs)*. Brownfield machinery has almost no monitoring, whereas CPSs have a large amount of sensors to monitor each component and condition separately. Consequently, there may be an information overload on CPSs [17], whereas there is almost no live information available on brownfield machinery [18]. In between, there is brownfield machinery that is enhanced with internet of things (IoT) devices [19] to monitor certain conditions of the production process. As a result, there is a diverse amount and granularity of information available in current factories—some over–fulfill, some under-fulfill the information needed for the fault diagnosis process.

For the fault diagnosis process, information is key to excavating the reason for a fault. The faster the problem is solved, the less costly is the production downtime. Fig. 3 shows the smart factory laboratory at Darmstadt University of Applied Sciences, Germany, where new innovations can be tested with the latest automation hardware in real-world scenarios. The smart factory produces fully functional electric relays that are used, e.g., in wind turbines. A high-bay storage with a three-axis robot lifts the unassembled relay onto a shuttle monorail system that interconnects all stations. The shuttle passes a six-axis robot that assembles the relay. Next, the pneumatic press assures connectivity between the relay and its socket. Additionally, in two inspection stations, the functionality of the relay is tested. One inspection is performed optically and by weight, the other electrically. Finally, the functioning relay will be stored back in the high-bay storage.

[4] https://www.hochschulkompass.de/en/ (accessed 4/11/2020).

[5] https://link.springer.com/referenceworkentry/10.1007%2F978-1-4899-7993-3_942-2.

[6] https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925 (accessed 4/11/2020).

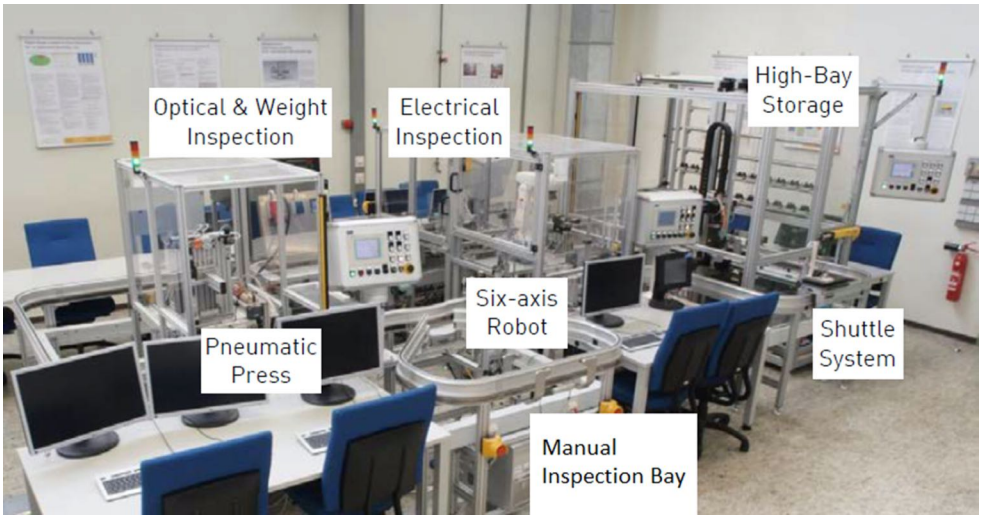


Fig. 3 Smart factory at Darmstadt University of Applied Sciences, Germany [20]

In such a complex environment, machine intelligence can assist the personnel in various areas to mitigate the consequences of a fault. We have published examples of automatic information extraction and information fusion to speed up the fault diagnosis process [21, 22]. Furthermore, we used autoencoder neural networks to find outliers in log data that a domain expert can use to start analyzing. In addition, we developed a semantic fusion process that is capable of semantically enriching machine events and fusing them together with documentation that guides the personnel throughout the fixing process. Guidance, or enablement of personnel, is key to solving the issues in such a complex environment. As complex the process of fault diagnosis can become, so manifold are the research directions currently in focus:

- Information provision (e.g., through the enhancement of brownfield machinery [23, 24])
- Formalization of machines, networks, and interactions, e.g., in cyber-physical systems of systems [25]
- Automatic information extraction and annotation, e.g., using machine learning to cope with big data [26]
- Assistance through intelligent visualisation, e.g., to visually pin-point faults and preconditions [27]

Addressing these research directions, we published a visual analytics (VA) model [28]. This combines production systems and their environment with computational models and visualizations to assist analysts in their daily tasks. We introduced and formalized the context as a standalone entity. We believe this will be an important entity in the fault diagnosis of the future. Consequently, we use our VA model for context-aware fault diagnosis. In addition, we published an article on a first dataset containing contextual faults to highlight the importance of the context and contextual information [20].

To conclude, analyzing the context of a fault will become more important in the future, since the complexity of smart factories will continue to increase.

Intelligent information systems in industrial applications

An example of an intelligent information system in industrial applications supports the production process of an automotive supplier for turbochargers. A system of this kind has been implemented [29] in accordance with ISO 18828-2 standard [30]. In this use-case, requirements such as customer-specific products and services, as well as lot-size one production, play a central role. In order to enable lot-size one production, production planning must be carried out and manufacturing needs to be transformed. Dynamically operating production lines need to react to changing demands in the shortest possible time. Outsourced production steps need to be converted to manufacturing networks with an integrated flow of information, data storage, and data access.

In order to meet these requirements, the *Knowledge Production Planning (KPP*[7]*) approach* ([31], see Fig. 4) with its KPP production planning ontology [32] offers the possibility to represent and store data sets in *Labeled Property Graphs (LPG*; see the section on AI technology below). KPP implementation is, among others, based on graph database Neo4j[8] platform [33], which enables graph search with *Cypher Query Language (CQL*[9]).

[7] Knowledge Based Production Planning (KPP)—Scheme and proof-of-concept implementation, https://kpp.fernuni-hagen.de, (accessed 28/10/2020).

[8] Neo4J Platform, https://neo4j.com (accessed 28/10/2020).

[9] Cypher Query Language https://neo4j.com/developer/cypher-query-language (accessed 28/10/2020).

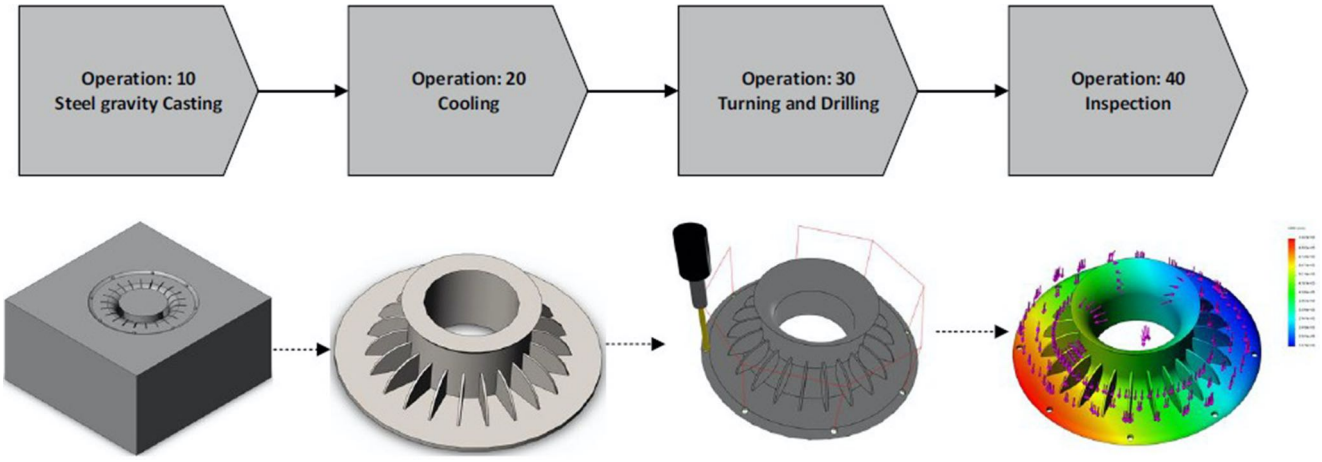


Fig. 4 Knowledge production planning process in component manufacturing of a turbocharger housing cover

Resource description framework (RDF) triple stores and LPGs both allow connected data to be explored and graphically depicted. But their models are different [34], and each have different strengths in different use cases. This has been shown in production planning use cases [35], in the fields of logistical production planning, additive production planning, and assembly production planning.

The RDF specification and model is more focused on data exchange, while LPGs are about data storage and querying. Another difference is that RDF does not offer any internal structures for nodes and edges. In contrast, LPG has these internal structures and thus enables a more detailed representation and annotation of nodes and edges. In LPGs, entities are called nodes with a unique identifier, plus a set of properties for characterization. Relationships between these nodes also have unique identifiers (IDs). Especially in LPGs, it is necessary to uniquely identify relationships to add a type and/or a set of key-value pairs in order to characterize them. "The important thing to remember here is that both the nodes and relationships have an internal structure, which differentiates this model from the RDF model" [34].

Building on this semantic formalization, and going beyond by using property graphs, KPP supports the collection, representation, administration, and reuse of knowledge related to production planning processes. In this way, KPP combines both a knowledge and a process perspective. Therefore, activities of a process with resources such as expert knowledge and documents can be commented on. These technologies offer a clearly defined formal semantic representation that supports the formal description of machine-readable production knowledge in industrial applications.

AI methodology

Knowledge-based AI (a.k.a. *symbolic AI*) and *machine learning* (a.k.a. *non-symbolic AI* or *subsymbolic AI*) are the two major AI approaches that complement each other. Engineering AI applications requires sound methodological skills. In this section, we present some selected aspects.

An ontology for machine learning

Machine learning (ML) is considered the most dominant AI approach today. Engineering professional ML applications is difficult and can be considered an art. Sound experience is mandatory and agreed-on engineering guidelines are scarce. ML libraries like scikit-learn[10] or TensorFlow[11], as well as integrated tools like RapidMiner[12] or Knime[13], offer hundreds of different ML approaches to choose from. Developers of ML applications are confronted with questions like:

- Which approaches for classification can be recommended with little training data?
- Which approaches for regression are able to deal with missing data? What are the approaches for filling missing data?
- Which prediction performance measures for classification can be recommended with unbalanced datasets?

Providers of integrated ML tools and ML libraries have started addressing the need for guidance via so-called *ML cheat sheets*. Examples include SciKit Learn—Choosing

[10] https://scikit-learn.org.

[11] https://www.tensorflow.org.

[12] https://rapidminer.com.

[13] https://www.knime.com.

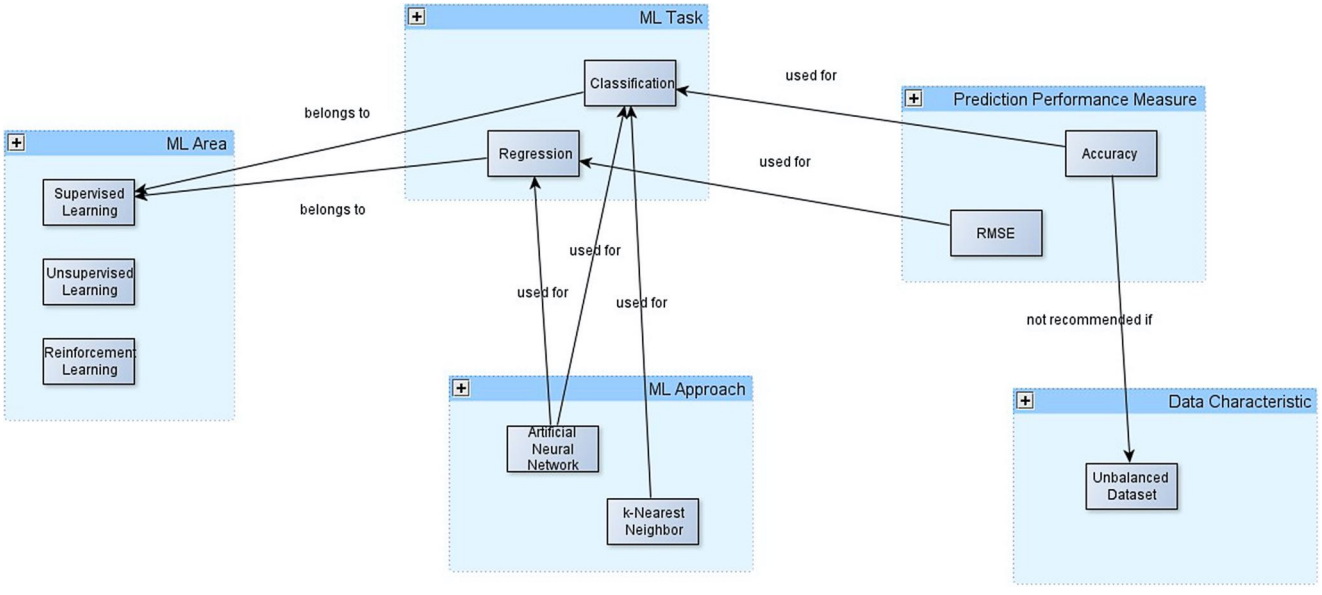


Fig. 5 Example concepts of a machine learning (*ML*) ontology

the right estimator[14], Microsoft Azure Machine Learning Algorithm Cheat Sheet[15], and Cheat Sheet: Machine Learning with KNIME Analytics Platform[16]. ML cheat sheets provide an overview of major ML concepts and how they interrelate. As such, they are simple ontologies, i.e., formal models of concepts and their relationships. However, they are usually restricted to one page in order to remain manageable. They may be a good guide for beginners, but cannot give a comprehensive overview of ML concepts. For detailed questions, ML engineers are obliged to search for proper documentation or follow a trial–error approach.

Wouldn't it be handy to have an ML ontology that makes it possible to answer questions like those mentioned above? This would support ML engineers when designing ML applications. It could also be used for teaching ML. In the future, such an ontology could also be used as a knowledge base for AI applications, e.g., to support automated orchestration of ML applications (auto-modelling) or a chat bot in answering questions about ML.

First attempts at ML ontologies have been undertaken: ML-Schema[17] provides a schema for ML ontologies to support interoperability between concrete ML ontologies. Its focus is on ML experiments, processing concrete datasets with concrete ML implementations. Overarching concepts like supervised/unsupervised/reinforcement learning are not in focus. Other attempts include OntoDM[18], Exposé[19], DMOP[20], and The MEX vocabulary[21]. However, none of these fully meets the requirements for an ML ontology as outlined above.

Therefore, we formed a working group to develop such an ML ontology, potentially by interlinking existing ontologies. Fig. 5 shows some example concepts of such an ML ontology.

This example expresses that the ML approach k-nearest neighbor can be used to classify tasks that belong to the area of supervised learning. Accuracy can be used as a prediction performance measure for classification tasks. Its use is not recommended if the data set is unbalanced. Artificial neural networks can be used for classification tasks, as well as for regression tasks—also belonging to the area of supervised learning. Root mean squared error (RMSE) can be used as a prediction performance measure for regression tasks.

The working group for building an ML ontology is, at the time of writing, in the phase of specifying goals of an ML ontology and researching related work. If you are interested in participating, please contact Bernhard Humm <bernhard.humm@h-da.de>.

[14] https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (accessed 20/11/2020).

[15] https://docs.microsoft.com/de-de/azure/machine-learning/media/algorithm-cheat-sheet/machine-learning-algorithm-cheat-sheet.svg (accessed 20/11/2020).

[16] https://www.knime.com/sites/default/files/110519_KNIME_Machine_Learning_Cheat%20Sheet.pdf (accessed 20/11/2020).

[17] http://ml-schema.github.io.

[18] http://www.ontodm.com.

[19] https://www.openml.org/downloads/expose.owl.

[20] https://raw.githubusercontent.com/agnieszkalawrynowicz/dmop.

[21] http://mex.aksw.org.

Recommendations for sound ontology modeling

Over the last 20 years, numerous recommendations on ontology modeling and meta-modeling, e.g., [36, 37], and even on bad practices [38], have been published. Criteria for sound ontologies are completeness, semantic correctness, interoperability, scalability, freedom from redundancy, compatibility to standards like RDF/OWL, and comprehensibility, e.g., by graph visualization (GV) [39].

To date, there are no truly universally accepted standards for sound ontology modeling. There is, e.g., an ongoing debate on how many levels of abstraction in a meta-model should be represented. While RDF seems to represent only one layer (everything is a resource), other approaches, including unified modeling language (UML), assume four layers (M0, ... , M3), while alternatively one could differentiate between the two layers: schema layer (SL, layer of classes and meta-classes) and the layer of individuals (IL).

It is recommended that ontologies be modeled step by step: identify basic concepts (classes), data properties (intrinsic), object properties (extrinsic), define axioms. This can be performed in analogy to agile approaches to software engineering.

As in software engineering, proper naming is important in order to avoid ambiguities and increase comprehensibility of models. Guidelines for naming are available. In [40], the author proposes using different special character prefixes for the identifiers of concepts like classes, data properties, object properties, processes, and relators. This makes it possible to automatically assign different colours and shapes for the GV of ontologies. For naming of object properties, the use of nouns rather than verbs is recommended. This allows for automatic derivation of the name of an inverse property, e.g., has_employee and is_employee_of.

The use of design patterns like the materialization pattern should be enforced. Also, (RDF) reification plays an important role in modeling knowledge artifacts. Reification makes it possible to model arbitrarily nested unasserted information about situations like beliefs, wishes, intentions, etc.

The following requirements for meta-modeling formulated in [41, 42] can also be regarded as modeling recommendations. The dual facet behavior of classes should be supported, i.e., a class can be regarded as a subclass and an instance of another class at the same time. Relationships between classes and individuals should be allowed, e.g., Yo-Yo Ma is_Expert_of Violin. Dynamically adding types should be possible. Rules for the instantiation of types at different levels should be provided. Information concerning domain subjects should be described locally to avoid fragmentation and redundancy. To work with multi-level models, support for queries and navigation between levels is required.

Modeling errors should be avoided: creating cycles in hierarchies, modeling individuals as classes, or modeling classes as individuals. Reasoners like Pellet, KARMA, HermiT, etc., can help validate the syntactical correctness of ontologies.

Some members of the Dagstuhl workshop formed a working group to prepare state of art documents and a website on the topic of sound ontology modeling. If you are interested in participating, please contact Hermann Bense <hb@bense.com>.

AI technology

In this section, we focus on one aspect of knowledge based AI, namely technologies for implementing and visualizing ontologies.

RDF versus labeled property graphs

Resource Description Framework (RDF)[22] is a W3C[23] standard for data interchange on the web. The *Web Ontology Language (OWL)*[24] is a language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is part of W3C's semantic web technology stack, which includes RDF, RDFS[25], SPARQL[26], etc.

In recent years, *Labeled Property Graphs (LPG)* [34, 43] have become an important area of research and application within the semantic web community. This was mainly driven by the success of NoSQL databases like Neo4J[27]. In [43], the LPG model is defined as follows: "The labeled property graph model consists of a set of nodes V (sometimes called vertices or knowledge subjects) and edges E (sometimes called arcs or links). An edge is always related to exactly two nodes with a fixed direction from a start to an end node, defining the property graph as a directed graph. [...] Both, nodes and edges, can store a set of key-value pairs, called properties and nodes can be tagged with labels additionally. Neo4J refers to edges as relationships."

RDF and labeled property graphs both provide ways to explore and graphically depict connected data. But they are different and each has different strengths in different use cases [34].

RDF, RDFS, and OWL provide rich modeling features for the implementation of semantic web applications. On

[22] https://www.w3.org/RDF/.

[23] https://www.w3.org.

[24] https://www.w3.org/2001/sw/wiki/OWL.

[25] https://www.w3.org/2001/sw/wiki/RDFS.

[26] https://www.w3.org/2001/sw/wiki/SPARQL.

[27] https://neo4j.com.

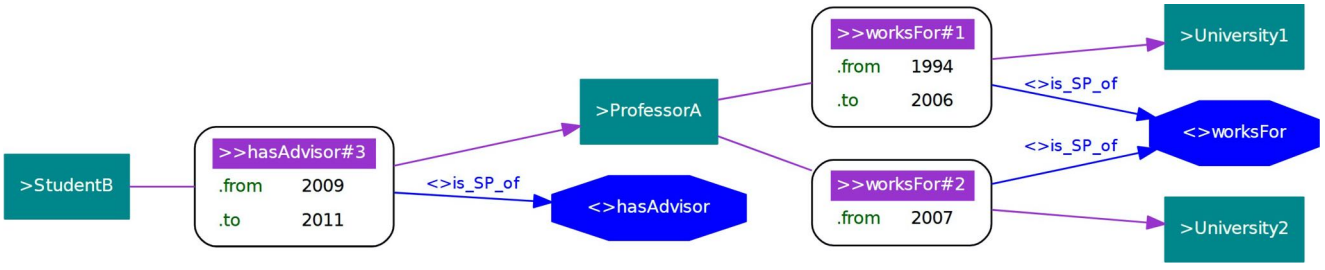


Fig. 6 Graph visualization of relationships between students, professors, and universities

the other hand, those technologies also require a lot of "work around modeling." RDF tuples do not allow for storing metadata properties like authors etc. [44]. Neither nodes nor edges have an internal structure. In contrast, LPG nodes and edges in Neo4J also have data properties, and nodes additionally have type information (called "labels" in Neo4J—not to be confused with labels in RDFS).

Graph visualization

The *graph visualisation* in Fig. 6 shows an example from the education context, namely the relationships between students, professors, and universities.

The graph visualization shows that the labels of directed relationships >>hasAdvisor and >>worksFor between the individuals >StudentB, >ProfessorA, >University1, and >University2 can be annotated with any number of data properties like.from and.to. This kind of modeling follows the singleton property approach (SPA) described in [45]. It is restricted to linking precisely one subject to one object.

If another participant like the contract for the employment needs to be incorporated, then one would have to use *n*-ary relations instead. An employment subject thus could have any number of object properties like <>Employer, <>Employee and <>Contract etc. *N*-ary relations can be used to model knowledge subjects like marriage, purchase, medical treatment, etc. See, e.g., the example about marriage relationships in Fig. 7.

The relator instances >MRG_BT1 and BRG_BT2 can have any number of object properties, to associate participants like partners husband and wife. Compared to direct relationships, where the first participant is always the subject and the second is the object of the relationship, none of the participants in an *n*-ary relation is privileged. This can be equally modeled in RDF and LPG.

In cases where there are exactly two participants being related by an object property, one can choose between the representation as direct relationships or *n*-ary relations. Typical examples when direct relationships should be used are object properties indicating a direction like <>next, <>knows, <>worksFor, <>hasAdvisor etc. Typical examples of when direct relationships should not be used include when the participants in a relationship are not privileged. This is, e.g., the case with the object properties <>Husband and <>Wife in a marriage, since this would afford the redundant annotation of properties like.from and.to within both vertices.

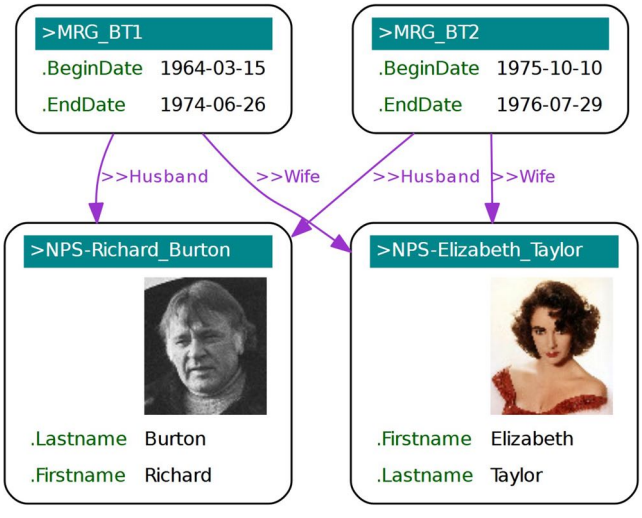


Fig. 7 Graph visualization of marriage relationships

Conclusions

AI applications are used every day. In this article, we present some AI applications from selected application domains, namely culture, education, and industrial manufacturing. Developing AI applications requires special skills. We have presented current trends, best practices, and recommendations regarding AI methodology and technology, focusing on ontologies (knowledge-based AI) and machine learning.

The selection of approaches presented is by no means comprehensive. It reflects a subset of topics that were discussed during the 2020 online Dagstuhl workshop on applied machine intelligence.

We will continue sharing our experiences in applied machine intelligence in Dagstuhl workshops and publish-

ing our results. If you work on intelligent applications in corporate contexts, you are cordially invited to participate in next year's workshop. Please contact: Bernhard Humm <bernhard.humm@h-da.de> or Thomas Hoppe <thomas.hoppe@htw-berlin.de>.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Ege B, Humm B, Reibold A (eds) (2015) Corporate Semantic Web – Wie Anwendungen in Unternehmen Nutzen stiften. Springer, Heidelberg (in German)
2. Hoppe T, Humm B, Reibold A (eds) (2018) Semantic applications – methodology, technology, corporate use. Springer, Berlin
3. Hoppe T (2020) Semantische Suche - Grundlagen und Methoden semantischer Suche von Textdokumenten. Springer Vieweg, Wiesbaden. ISBN 978-3-658-30426-3 (mit Beiträgen von Bernhard Humm)
4. Bense H, Gernhardt B, Haase P, Hoppe T, Hemmje M, Humm B, Paschke A, Schade U, Schäfermeier R, Schmidt M, Siegel M, Vogel T, Wenning R (2016) Emerging trends in corporate semantic web – selected results of the 2016 Dagstuhl workshop on corporate semantic web. Informatik Spektrum 39(6):474–480
5. Busse J, Humm B, Lübbert C, Moelter F, Reibold A, Rewald M, Schlüter V, Seiler B, Tegtmeier E, Zeh T (2015) Actually, what does "Ontology" mean? A term coined by philosophy in the light of different scientific disciplines. J Comput Inform Technol 23(1):29–41. https://doi.org/10.2498/cit.1002508
6. Hoppe T, Humm B, Schade U, Heuss T, Hemmje M, Vogel T, Gernhardt B (2015) Corporate semantic web – applications, technology, methodology. Informatik Spektrum 39(1):57–63. https://doi.org/10.1007/s00287-015-0939-0
7. Humm BG, Bense H, Classen M, Geißler S, Hoppe T, Juwig O, Paschke A, Schäfermeier R, Siegel M, Weichhardt F, Wenning R (2019) Current trends in applied machine intelligence. Informatik Spektrum 42(1):28–37. https://doi.org/10.1007/s00287-018-01127-0
8. Schade U, Fillies C, Humm B, Reibold A, Schumann F, Weichhardt F 5 Years of Semantics Workshops in Schloss Dagstuhl: it connects! https://www.researchgate.net/publication/336798165_5_Years_of_Semantics_Workshops_in_Schloss_Dagstuhl_it_connects_4_Day_Workshop_-30_People_-1_Common_Cause German translation: https://www.researchgate.net/publication/335621540_5_Jahre_Semantik-Workshops_im_Schloss_Dagstuhl_das_verbindet. Accessed 28 Oct 2020. https://doi.org/10.13140/RG.2.2.17594.54722
9. Thull B, Diwisch K, Marz V (2015) Linked Data im digitalen Tanzarchiv der Pina Bausch Stiftung. In: Ege B, Humm B, Reibold A (eds) Corporate Semantic Web – Wie Anwendungen in Unternehmen Nutzen stiften. Springer, Heidelberg (in German)
10. Thull B (2018) Entwicklung experimenteller digitaler Archive auf Basis von Linked Data-Standards. In: Schneider W, Fülle H, Henniger C (eds) Performing the Archive. Theaterpolitik für ein Archiv des Freien Theaters. Hildesheimer Universitätsschriften, vol 34, pp 245–276 (in German)
11. Stiftung zur Förderung der Hochschulrektorenkonferenz (2019) Studieren und promovieren in Deutschland. Informationen über deutsche Hochschulen, Studiengänge, Promotionen. https://www.hochschulkompass.de/home.html. Accessed 30 Jan 2020
12. Berners-Lee T, Hendler J (2001) Publishing on the semantic web. Nature 410:1023–1024
13. Blois M, Escobar M, Choren R (2007) Using agents and ontologies for application development on the semantic web. J Braz Comp Soc. https://doi.org/10.1007/BF03192408
14. Berners-Lee T et al (2001) The semantic web. Sci Am 284(5):34–43 (www.jstor.org/stable/26059207. Accessed 30 Oct. 2020.)
15. Shadbolt N, Berners-Lee T, Hall W (2006) The semantic web revisited. IEEE Intell Syst 21:96–101
16. Nawroth C, Schmedding M, Brocks H, Kaufmann M, Fuchs M, Hemmje M (2015) Towards cloud-based knowledge capturing based on natural language processing. In: Proceedings of CloudForward Conference Pisa PCS. vol 2015. Elsevier, Vienna, Austria
17. Xu X, Hua Q (2017) Industrial big data analysis in smart factory: current status and research strategies. IEEE Access. https://doi.org/10.1109/access.2017.2741105
18. Etz D, Brantner H, Kastner W (2020) Smart manufacturing retrofit for Brownfield systems. Proc Manuf. https://doi.org/10.1016/j.promfg.2020.02.085
19. Strauss P et al (2018) Enabling of predictive maintenance in the Brownfield through low-cost sensors, an IIoT-architecture and machine learning. 2018 IEEE International Conference on Big Data (Big Data). https://doi.org/10.1109/bigdata.2018.8622076
20. Kaupp L, Webert H, Nazemi K, Humm BG, Simons S (2020) CONTEXT: An industry 4.0 dataset of contextual faults in a smart factory (in press). In: Longo F, Affenzeller M, Padovano A (eds) Proceedings of the International Conference on Industry 4.0 and Smart Manufacturing (ISM). Elsevier, Vienna, Austria
21. Beez U et al (2018) Context-Aware Documentation in the Smart Factory. Semantic Appl. https://doi.org/10.1007/978-3-662-55433-3_12
22. Kaupp L, Beez U, Hülsmann J, Humm BG (2019) Outlier detection in temporal spatial log data using autoencoder for industry 4.0. In: Macintyre J, Iliadis L, Maglogiannis I, Jayne C (eds): Engineering Applications of Neural Networks. Springer, Cham, pp 55–65. https://doi.org/10.1007/978-3-030-20257-6_5
23. Ratasich D et al (2019) A roadmap toward the resilient Internet of things for Cyber-physical systems. IEEE Access. https://doi.org/10.1109/access.2019.2891969
24. Wan J et al (2016) Software-defined industrial Internet of things in the context of industry 4.0. IEEE Sensors J. https://doi.org/10.1109/jsen.2016.2565621
25. Engell S et al (2015) Core research and innovation areas in cyber-physical systems of systems. Cyber Phys Syst Des Model Eval. https://doi.org/10.1007/978-3-319-25141-7_4
26. Shafique M et al (2020) Robust machine learning systems: challenges, current trends, perspectives, and the road ahead. IEEE Des Test. https://doi.org/10.1109/mdat.2020.2971217
27. Zhou F et al (2019) A survey of visualization for smart manufacturing. J Vis 22(2):419–435. https://doi.org/10.1007/s12650-018-0530-2
28. Kaupp L, Nazemi K, Humm BG (2020) An industry 4.0-ready visual analytics model for context-aware diagnosis in smart manufacturing. In: Proceedings of the 24th International Conference In-

formation Visualisation (IV) IEEE, pp 337–346. https://doi.org/10.1109/IV51561.2020.00064
29. Kugler P (2019) Analyse, Vergleich und Evaluation eines auf SAP basierenden Produktionsplanungssystems gegenüber der Methode "Knowledge-based Production Planning" sowie die Entwicklung einer geeigneten semantischen Schnittstelle. FernUniversität, Hagen (Abschlussarbeit)
30. ISO 18828-2 (2016) Industrial automation systems and integration -- Standardized procedures for production systems engineering -- Part 2: Reference process for seamless production planning, ISO, Geneva, Switzerland
31. Gernhardt B, Vogel T, Hemmje M (2017) Knowledge-based production planning for industry 4.0. In: Hoppe T, Humm B, Reibold A (eds) Semantic applications. Springer Vieweg, Berlin, Heidelberg, pp 181–202
32. Klaus M (2020) Erweiterung und Implementierung eines Ontologie-Modells und der semantischen Funktionalitäten einer verteilten und kollaborativen Unterstützung für die Produktionsplanung auf Basis von Neo4j sowie deren exemplarische Evaluation. FernUniversität, Hagen (Abschlussarbeit)
33. openCypher Project (2017) https://www.opencypher.org/. Accessed 28 Oct 2020
34. Barrasa J (2017) RDF triple stores vs. Labeled property graphs: what's the difference? Aug 18, 2017. https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference. Accessed 24 Nov 2020
35. Vogel T, Gernhardt B, Hemmje M (2020) The KPP ecosystem and its smart production planning services, workshop on management for industry 4.0. IEEE/IFIP Network Operations and Management Symposium (NOMS2020), Budapest
36. Noy NF, Mcguinness DL (2001) Ontology Development 101: A Guide to Creating Your First Ontology. https://www.researchgate.net/publication/243772462_Ontology_Development_101_A_Guide_to_Creating_Your_First_Ontology. Accessed 24 Nov 2020
37. Schulz S, Seddig-Raufie D, Grewe N, Röhl J, Schober D, Boeker M, Jansen L (2012) Guideline on developing good ontologies in the biomedical domain with description logics, version 1.0. http://www.purl.org/goodod/guideline. Accessed 24 Nov 2020
38. Smith B (2005) Against fantology. In: Marek J, Reicher EM (eds) Experience and analysis. öbv&hpt, Vienna, pp 153–170 (http://ontology.buffalo.edu/bfo/Against_Fantology.pdf . Accessed 24/11/2020)
39. Gansner ER, Koutsofios North ES (2015) Drawing graphs with dot. dot user's manual. http://www.graphviz.org/pdf/dotguide.pdf (Created 5 Jan 2015). Accessed 24 Nov 2020
40. Bense H (2014in) The unique predication of knowledge elements and their visualization and factorization in ontology engineering. In: Garbacz P, Kutz O (eds) Formal ontology in information systems Proceedings of the Eighth International Conference (FOIS 2014), Rio de Janeiro, 22-25.09.2014 IOS Press, Amsterdam. ISBN 978-1-61499-437-4
41. Guizzardi G, Almeida JPA, Guarino N, Carvalho VA (2015) Towards an ontological analysis of powertypes. International Workshop on Formal Ontologies for Artificial Intelligence (FOFAI).
42. Neumayr B, Grün K, Schrefl M (2009) Multi-level domain modeling with M-objects and M-relationships. Proceedings of conceptual modelling 2009, sixth asia-pacific conference on conceptual modelling (APCCM 2009), Wellington, New Zealand, January 20-23 2009. https://www.researchgate.net/publication/220268481_Multi-Level_Domain_Modeling_with_M-Objects_and_M-Relationships. Accessed 24 Nov 2020
43. Lekschas F (2015) What is a labeled property graph? 2 Sep 2015. https://github.com/flekschas/owl2neo4j/wiki/What-is-a-labeled-property-graph%3F. Accessed 24 Nov 2020
44. Gandon F, Corby O (2010) Name That Graph or the need to provide a model and syntaxextension to specify the provenance of RDF graphs. W3C Workshop - RDF Next Steps , Jun2010, Palo Alto, United States.⟨http://www.w3.org/2009/12/rdf-ws/⟩.⟨hal-01170906. https://www.researchgate.net/publication/282066423_Name_That_Graph_or_the_need_to_provide_a_model_and_syntax_extension_to_specify_the_provenance_of_RDF_graphs. Accessed 24 Nov 2020
45. Nguyen V, Bodenreider O, Thirunarayan K, Fu G, Bolton E, Rosinach NQ, Furlong LI, Dumontier M, Sheth A (2015) On reasoning with RDF statements about statements using singleton property triples. arXiv:1509.04513 [cs.AI] (24/11/2020)

Informatik Spektrum (2021) 44:115–121
https://doi.org/10.1007/s00287-020-01323-x

HAUPTBEITRAG

Fernlehren und Fernlernen von Objektorientierter Programmierung (OOP)

Mathias Ellmann[1]

Online publiziert: 13. November 2020

Zusammenfassung
Dieses Papier beschreibt unsere Lehrerfahrungen in einer virtuellen Lehr- und Lernumgebung für eine Vorlesung über objektorientierte Programmierung (OOP) in Java. Wir verwenden die Live-Meeting-Anwendung Adobe Connect sowie Lern- und Lehrmethoden für OOP-Vorlesungen wie Online-Coding, Online-Debugging, Online-Gruppenräume, Online-White-Boards und Online-Fragebögen, um unsere Lehr- und Lernziele zu messen und zu erreichen. Wir haben festgestellt, dass Online-Coding in Kombination mit Online-Debugging oder einem Codebeispiel das mit einem Kommentar versehen ist, in dieser virtuellen Lehr- und Lernumgebung am effektivsten funktioniert, um den Studierenden ein besseres Verständnis der Programmierkonzepte und -methoden zu vermitteln. Online-Coding und Online-Debugging sollten unmittelbar nach der Vermittlung der OOP-Konzepte und -Methoden durchgeführt werden. Andere Lehrmethoden wie das Unterrichten der OOP-Konzepte ohne ein konkretes Beispiel oder eine Anwendung führen zu Frustration und Unzufriedenheit bei den Studierenden.

Einleitung

Heutzutage wird der Unterricht mehr und mehr online angeboten [1, 2]. Es ist möglich, einen kompletten Master- oder Bachelor-Studiengang in einem MOOC (Massive Open Online Course) [3] wie in coursera[1] oder MOODLE [4, 5] durchzuführen. Studien hatten eine hohe Abwanderungsrate von MOOCs gezeigt, z. B. wegen der fehlenden Gelegenheiten, online mit einem Dozenten oder anderen Studierenden zu sprechen [6]. Infolgedessen fühlten sich die Studierenden beim Studieren und Zuhören des Dozenten isoliert. Eine andere Möglichkeit, online in einer virtuellen Umgebung zu unterrichten, ist ein Videokonferenzsystem für E-Learning-Zwecke, in das sich die Studierenden integriert und nicht isoliert fühlen [6]. Ein Beispiel für ein solches E-Learning-System ist Adobe Connect[2], in dem eine Online-Diskussion mit einem Dozenten und Gruppenräumen möglich ist. Diese Umgebung bietet viele Möglichkeiten, online zu lehren und die Vorlesungen einzurichten, indem sie die effektivsten Phasen (z. B. Üben durch Ausprobieren, andere unterrichten oder Gruppendiskussionen [7]) in der typischen Lernpyramide [7, 8] abdeckt und den Studierenden Feedback gibt, um bessere Lernergebnisse zu erzielen [9, 10].

Missverständnisse in Bezug auf das Verständnis der OOP-Terminologie und -Konzepte sind ein bekanntes Problem [11]. Mit den Möglichkeiten, die Adobe Connect bietet, um die Studierenden in den Lernprozess einzubeziehen und ihnen Feedback [9, 10, 12] im gleichen Lehr- und Lernkontext zu geben, wollen wir die Missverständnisse vermeiden und ihre Bedürfnisse im Lernprozess [9] während der Vorlesung in der OOP abdecken. Zudem können Sie sich durch das Lernen im Kontext des zukünftigen Arbeitskontextes – der virtuellen Umgebung- leichter an die Inhalte zu OOP erinnern [13], da Sie sich wieder im demselben Kontext wie im Lernkontext befinden [2, 13].

Viele Studierende beschreiten Ihr Studium heutzutage nebenberuflich [14]. Ein effektives Anwenden der heute bekannten Lehr- und Lernmethoden [15] ist heute umso wichtiger, sodass die Studierenden, weniger belastend die komplexen Inhalte insbesondere im Objektorienten Programmieren erlernen können und diese im Arbeitskontext handlungsfähig sind.

[1] https://de.coursera.org.

[2] https://adobeconnect.eu.

✉ Mathias Ellmann
mathias.ellmann.cs@gmail.com

[1] DIPLOMA University of Applied Sciences, Bad Sooden-Allendorf, Deutschland

Kurs-Aufbau

Einzelheiten zum Kurs

Die Modulvorlesung in objektorientierter Programmierung (OOP) wird in einem zweisemestrigen Kurs an der DIPLOMA Fachhochschule im Studiengang Wirtschaftsinformatik gehalten. In einem Semester wird der Kurs Grundlegende Programmiertechniken und im nachhergehenden Semester wird der Kurs Objektorientiertes Programmieren gehalten.

Im Kurs Grundlegende Programmiertechniken lernen die Studierenden grundlegende Methoden der objektorientierten Programmierung in Java kennen. Die Lehrveranstaltung behandelt die Grundelemente von Java wie Kontrollstrukturen, Klassen und Vererbung sowie Algorithmen wie Bubble Sort. Im Kurs Objektorientierte Programmierung werden ausgewählte API-Elemente in Java wie Netzwerkverbindungen, Sockets oder UI-Programmierung mit dem Abstract Window Toolkit (awt) behandelt. Zu jedem Kurs gibt es ein Kursbuch von ca. 80 Seiten mit Beispielen und Übungen, die speziell für das Selbststudium der objektorientierten Programmierung geschrieben sind.

In der Vorlesung verteilten wir Vortragsfolien, die ein OOP-Thema aus dem Kursbuch zusammenfassten und Online-Übungen zu einem Thema als Kontrollstrukturen in OOP enthielten. Wir diskutierten jedes Thema unter Verwendung von Adobe Connect und seinen Lehr- und Lernfunktionen. Die Präsenzzeit jeder Vorlesung beträgt etwa 20 Stunden (insgesamt für das OOP-Modul 40 Stunden), so dass die Online-Vorlesung das Selbststudium der Studierenden erweitert. In der Regel treffen sich die Studierenden mit dem Dozenten am Samstag von 09:30–12:45 Uhr oder nachmittags von 13:15–16:30 Uhr (in der Regel in einem Block von insgesamt 4–5 Wochenenden nacheinander).

Durch die Kombination von Studienbüchern und der virtuellen Lehr- und Lernumgebung können alle Aspekte der Bloom-Taxonomie [16] abgedeckt werden, wie Tab. 2 zeigt. Das Studienbuch enthält Übungen, die die Studierenden innerhalb des Semesters lösen können. Die Dozentin oder der Dozent stellt Übungen in den Gruppenräumen zur Verfügung. Die Studierenden können ihre Aufgabe am Whiteboard lösen und die Lösungen sowohl online mit der Gruppe als auch mit dem Dozenten diskutieren. Die Studierenden können auch ihre Lösungen aus den Übungen im Studienbuch mit der Lösung der Übung vergleichen und analysieren. Die Studenten können die Lösungen in der Live-Coding, in der Online-Debugging-Sitzung sowie in den Code-Beispielen analysieren. Die Studierenden können das Wissen über OOP in ihrer IDE oder in der virtuellen Umgebung in den Gruppenräumen, in der Live-Codingsitzung und am Whiteboard anwenden. Die Studierenden führen einen Selbsttest durch, in dem sie ihr Verständnis und ihre Erinnerung an den OOP-Inhalt im Online-Fragebogen unter Beweis stellen.

Tab. 1 Verwendete Pods (Lehr- und Lernfunktionen) in der Adobe Connect-Anwendung

Pod	Beschreibung und Verwendung
Filesharing	Teilen und Besprechen von Folien eines OOP-Themas
Whiteboard	Zeichnen z. B. UML-Modelle online
Bildschirm freigeben	Gemeinsamer Bildschirm für Online-Coding und Online-Debugging
Video des Dozenten	Virtuelle Kommunikation mit anderen Studierenden
Gruppenraum	Zusammenarbeit mit Studierenden
Chat für Fragen	Diskussion mit Studierenden
Online-Fragen	Online-Fragebogen, um z. B. den Lernfortschritt zu messen
Weblink	Studenten in ihrem Browser zu einer API-Referenzdokumentation führen

Studenten

Die Lerngruppe besteht in der Regel aus 6–10 Studierenden. Die Studierenden haben seit 1–2 Jahren Erfahrungen mit Informatik-Themen (sie haben bereits die Programmiersprache C++ gelernt). Die Studierenden haben in der Regel einen starken technischen Hintergrund, aber nicht konsequenterweise in Informatik als im Maschinenbau als Techniker. Die Studierenden studieren samstags in Teilzeit und nebenberuflich nach einer 35–40-Stunden-Woche. Alle Vorlesungen werden online in einer virtuellen Lehr- und Lernumgebung gehalten, in der die Studierenden mit ihren Videokameras und Mikrophonen anwesend sind. Die Studierenden verbinden sich in der Regel zu Hause mit dem Dozenten.

Infrastruktur

Für diesen Kurs verwenden wir die Live-Meeting-Anwendung Adobe Connect (siehe Abb. 1 und Abb. 2). Die Lehr- und Lernfunktionen in Adobe Connect werden als pods [17, 18] bezeichnet. Wir haben die Vorlesungsfolien online mit der File-Sharing-Funktion geteilt und während der Vorlesung mehrere Pods verwendet. Wir beschreiben kurz die Pods in Tab. 1.

Fernlehren und Fernlernen

Im Folgenden beschreiben wir unsere Erfahrungen nach vier Semestern Vorlesungszeit der Modulvorlesung OOP. Wir haben die Modulvorlesung Objektorientierter Programmierung vor acht Studentengruppen (zu jeweils 4 Gruppen in Grundlegende Programmiertechniken und Objektorien-

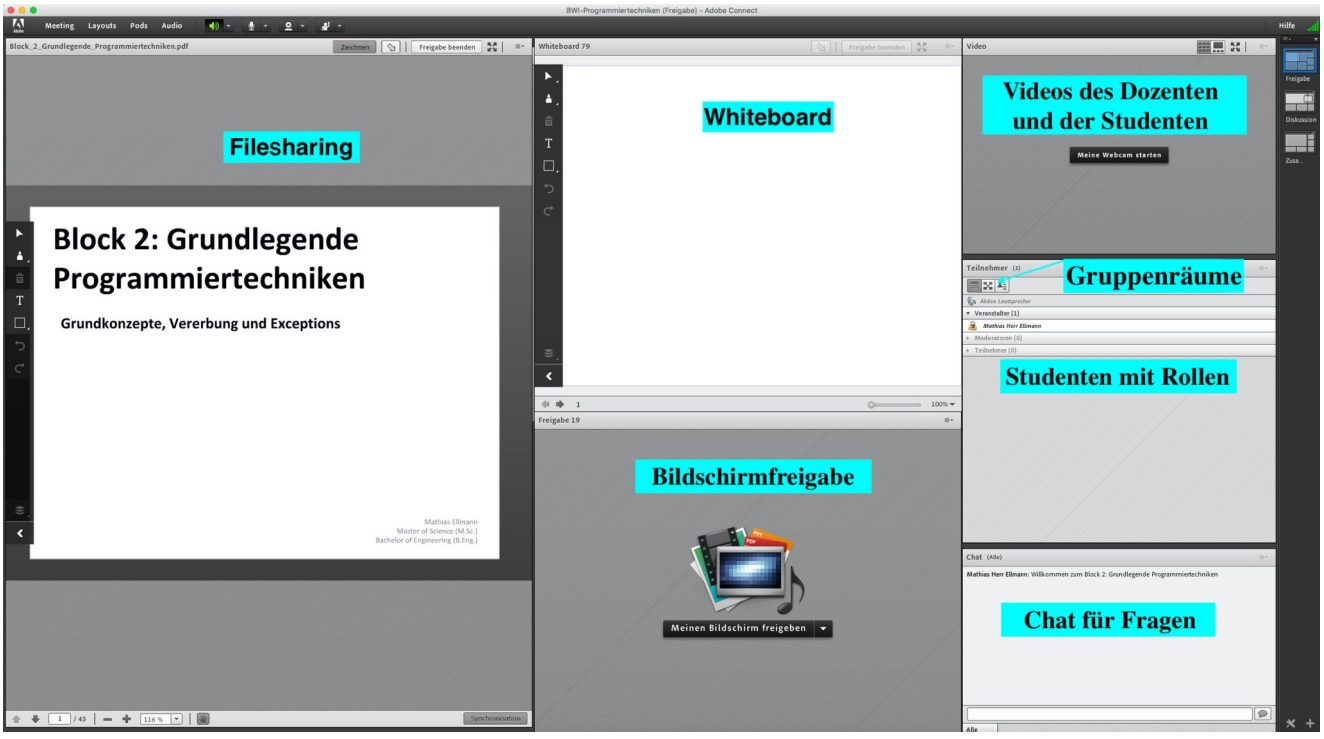


Abb. 1 Live-Meeting-Anwendung mit Adobe Connect

Tab. 2 Deckung der Bloom's Pyramide (neue Version) [16] mit Studienbuch und virtuellem Lehr- und Lernkontext [2]

Stage	Study book	Virtueller Lehr- und Lernraum
Creating	In-Buch-Übungen	Gruppenräume und Whiteboard
Evaluating	In-Buch-Lösungen	Online-Diskussionen
Analyzing	In-Buch-Lösungen	Live-Coding, Online-Debugging, Code-Beispiele
Applying	Eigene IDE	Gruppenräume, Live-Coding und Whiteboard
Understanding	Selbsttest	File-Sharing, Diskussionen mit Dozenten und Gruppenräumen
Remembering	Selbsttest	Fragebogen

tierte Programmierung; ca. 160 Stunden mit ca. 5–10 regulären Studierenden) gehalten. Zur Strukturierung unserer Lehre, verwendeten wir die Lernpyramide (siehe Abb. 3) und das Lernmodell von Race (siehe Abb. 4). Die Lernpyramide stellt ein Effektivitätsmaß je Lehr- und Lernmethode im Rahmen der Vorlesung dar. Die Lernkreise von Race stellen die Lernmethoden dar, welche für effektives Lernen durchlaufen werden.

Aktivierung

Wir stellten fest, dass Aktivierung und eine aktive Beteiligung der Studierenden sehr nützlich waren, um unsere Lehr- und Lernziele zu erreichen und Missverständnisse beim Verständnis der OOP-Terminologie und -Konzepte zu vermeiden [11]. Wir verwendeten verschiedene Methoden, um unsere Studierenden zu aktivieren und einzubeziehen. Wir aktivierten die Studierenden durch direkte Fragen und Diskussionen, Fragebögen (siehe Abb. 2) mit einer Frage und Mehrfachantworten. Die Online-Fragen und -Diskussionen waren in der Online-Umgebung sehr gut möglich. Die Studierenden nahmen sehr oft am Online-Fragebogen teil. Sie beteiligten sich auch sehr aktiv an der Diskussion über ihre Fehler.

Wir nutzten auch die Bildschirmfreigabe- oder Remote-Funktion zur Aktivierung der Studierenden während der Online-Vorlesung (siehe Abb. 1). Die Studierenden konnten in die Rolle eines Dozenten schlüpfen, damit sie anderen Studierenden zeigen konnten, wie sie ihr Programm programmieren (die Lehrmethode heißt umgekehrter Klassenraum [12, 20, 21]). Der Dozent konnte einige Änderungen am Code vornehmen oder den Code mit den Studenten debuggen.

Online-Coding

Die Online-Coding war die effektivste Lehrmethode im Hinblick auf die Evaluations- und Feedback-Phase des Kur-

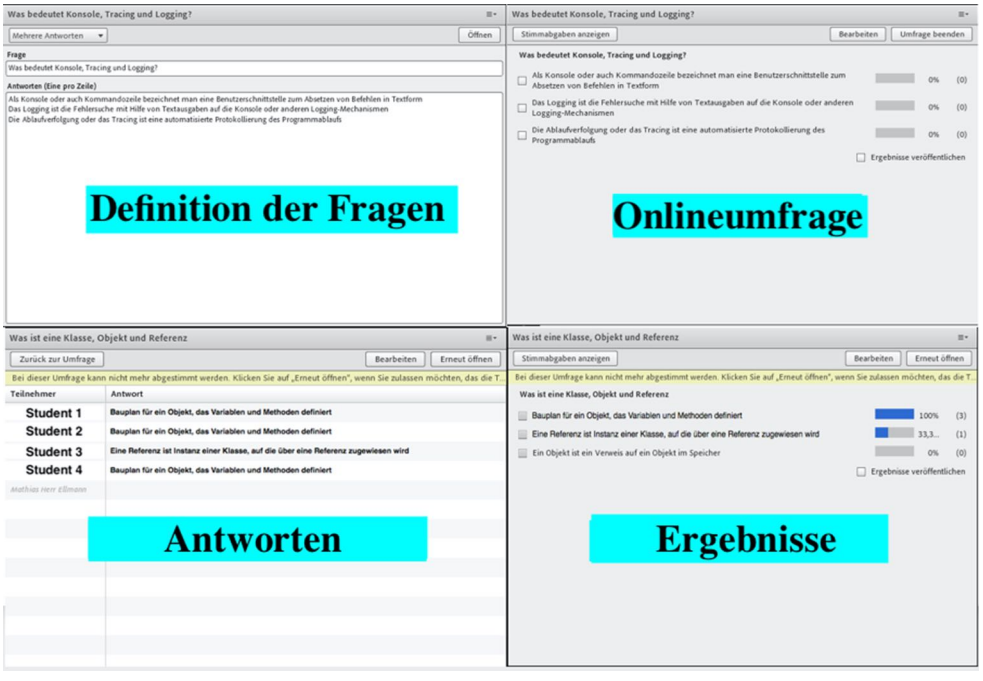


Abb. 2 Definitionen des Fragebogens und Antworten des Fragebogens mit Adobe Connect

ses am Ende des Semesters. Wir denken auch, dass die virtuelle Lehr- und Lernumgebung diese Methode ideal unterstützt, da der Dozent und die Studierenden den Lernkontext [2] nicht gewechselt haben. Die Studierenden verwenden in ihrem Betriebssystem (OS) die Lehrsoftware Adobe Connect und die Eclipse Integrated Development Environment (IDE)[3] für Entwickler im gleichen Lehr- und Lernkontext [2].

Wir haben Online-Coding in jedem OOP-Thema in der Vorlesung verwendet und unsere Code-Beispiele über den Debugger der Eclipse-IDE gedebuggt (siehe Abb. 5). Wir haben den Debugger verwendet, um zu zeigen, wie Zeichenketten gespeichert werden, z. B. an welcher Position ein Zeichen gespeichert wird. Wir haben den Debugger auch verwendet, um Zustände eines Objekts oder im Programm an einem bestimmten Zustand zu zeigen, sowie z. B. zu zeigen, wie sich der Datenstruktur-Stack in der Programmierumgebung verhält.

Die Studenten sagten, dass das Online-Coding sehr hilfreich war, wenn das Online-Coding direkt nach der Vermittlung eines Programmierkonzepts durchgeführt wurde. Den Studenten einfach zu erklären, wie API-Elemente zu verwenden sind, ohne dies in einer IDE zu zeigen, war sehr unbefriedigend und frustrierend für sie, da sie die Wirkung und das Ergebnis z. B. einer Methode oder eines Programmierkonzepts nicht verstehen konnten.

[3] https://www.eclipse.org/downloads/packages/release/2019-09/r/eclipse-ide-java-developers.

Gruppenräume

Wir bildeten Gruppen über die Gruppenfunktion von Adobe Connect, bei der wir die Schüler nach dem Zufallsprinzip in Gruppen mit einer Größe von 4–5 Mitgliedern unterteilten (siehe Abb. 1). Ihnen standen Pods als White-Board und andere zur Verfügung, um ein Programmierproblem zu lösen und ein Programmierkonzept, z. B. eine Klasse, zu visualisieren. Sie wurden mit Aufgaben wie dem Verständnis der Verwendung der API-Elemente oder eines kleinen Programms mit mehreren Klassen und Methoden konfrontiert und mussten die Lösung gemeinsam erarbeiten. Nach einem Gruppentreffen präsentierte jede Gruppe ihre Ergebnisse den anderen Gruppenmitgliedern. Wir fanden, dass diese Methode sehr nützlich war, um ein tieferes Verständnis der Programmierkonzepte zu erhalten.

Fragebögen

Eine Vorlesung dauerte 3 Stunden und 15 Minuten. In der Hälfte der Vorlesung (nach ca. zwei Stunden) machten wir eine Pause von 10 Minuten. Nach der Pause wurden den Studierenden mehrere Fragen zu den Vorlesungsinhalten zu den zwei Stunden der Vorlesung gestellt. Wir werteten die Antworten direkt aus und gaben ein positives Feedback (mit Lob und Anerkennung [22]). Wir konnten so die missverstandenen Konzepte gleich nach ihrer Erläuterung klären. Wir konnten auch verstehen, welche Konzepte der OOP in Java nicht verstanden wurden und intensiver behandelt werden mussten.

Chat für Fragen und Weblinks

Wir nutzten die Chat-Funktion von Adobe Connect während der Vorlesung als Gelegenheit, zusätzliche Links oder Informationen auszutauschen, die die Vorlesung erweiterten. Während der Online-Codingsitzungen hatten wir auch API-Methoden verwendet. Wir haben Sie mit den Studierenden gemeinsam nachgeschlagen, indem wir die Weblink-Funktion von Adobe Connect nutzten. Bei der Verwendung der Web-Link-Funktion von Adobe Connect wurden die Studierenden von ihrem Dozenten und Adobe Connect in ihrem Webbrowser zur entsprechenden Java-API-Website geleitet. So konnten die Studierenden lernen, wie man eine Online-Ressource für Entwicklungsprobleme zu dem Zeitpunkt nutzen kann, zu dem sie im Entwicklungskontext [23, 24] benötigt wird. Diese Episode [25] und wir als Modell [26] konnten dadurch eine Problemlösung mittels externer Ressource aufzeigen, was den Lerneffekt verstärkte und die Handlungsfähigkeit in solchen Problemkontexten förderte.

Zusammenfassung der Ergebnisse

Unsere Ergebnisse zur Wirksamkeit unseres Unterrichts orientierten sich an der Lernpyramide (siehe Abb. 3) und dem Modell von Race (siehe Abb. 4) und zeigten deren Effektivität auf. Wir versuchten wie in der Lernpyramide aufgezeigt, die Studierende direkt am Lernprozess teilzunehmen. Wie liesen die Studierende miteinander diskutieren und Sie konnten auch andere in der gleichen Gruppe mit Begleitung des Dozenten unterrichten. Insbesondere das Üben durch tun in diesem problemorientierten Fach, motivierte die Studierende, trotz der Komplexität des Faches OOP weiter die Inhalte aktiv zu studieren. Wie nach Race (siehe Abb. 4) wollten die Studierenden lernen, in OOP zu programmieren. Wir erklärten ihnen OOP-Konzepte und -Methoden, und lieferten ein OOP-Beispiel, das die Wirkung der angewandten Methoden und OOP-Konzepte aufzeigte. Wir gaben nach den Fragebögen Feedback, und die Studierenden konnten ihre Problemlösungen in den Gruppenräumen und in der Online-Diskussion mit dem Dozenten verbalisieren.

Abb. 3 Lernpyramide nach National Training Laboratories (NTL), Bethel Maine [7, 8]

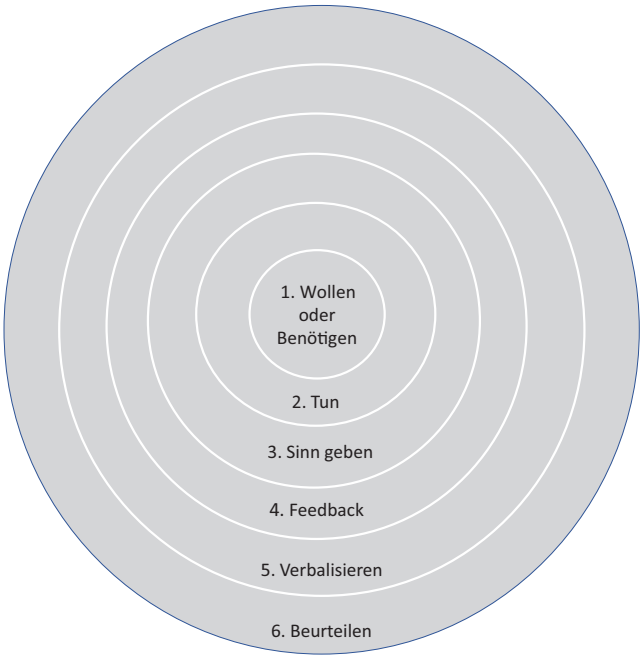


Abb. 4 Lernprozess nach Phil Race [10, 19]

Threats to Validity und Einschränkungen

Die Idee der Vorlesung besteht darin, ein Studienbuch für das Selbststudium zu erweitern. Bei anderen Vorlesungen, in denen OOP gelehrt wird und kein Studienbuch vorhanden ist, könnten unsere erklärten Methoden und Erfahrungen weniger Gültigkeit haben, da die Studierende vemittelte Inhalte nicht nochmals nachlesen können. Basierend auf den Ergebnissen der Lernpyramide [7] erwarten wir ähnliche Ergebnisse. Alle Studenten hatten einen starken technischen beruflichen Hintergrund, was auch die Ergebnisse in anderen OOP-Vorlesungen mit Studenten, die einen anderen Hintergrund haben, beeinflussen könnte.

Die gesamte Modulvorlesung zu OOP hat eine Länge von insgesamt 40 Stunden. Wir konnten im Rahmen dieser Zeit nur begrenzt auf alle Übungen im Studienheft Bezug nehmen und ein Feedback geben. Wir boten den Studierenden an, offenstehende Übungen und aufgetretende Probleme, die im Selbststudium des Übungsheftes aufkamen, in der Vorlesung zu besprechen. Wir fanden es sehr nützlich, diese Einschränkung mit Lernmöglichkeiten von On-

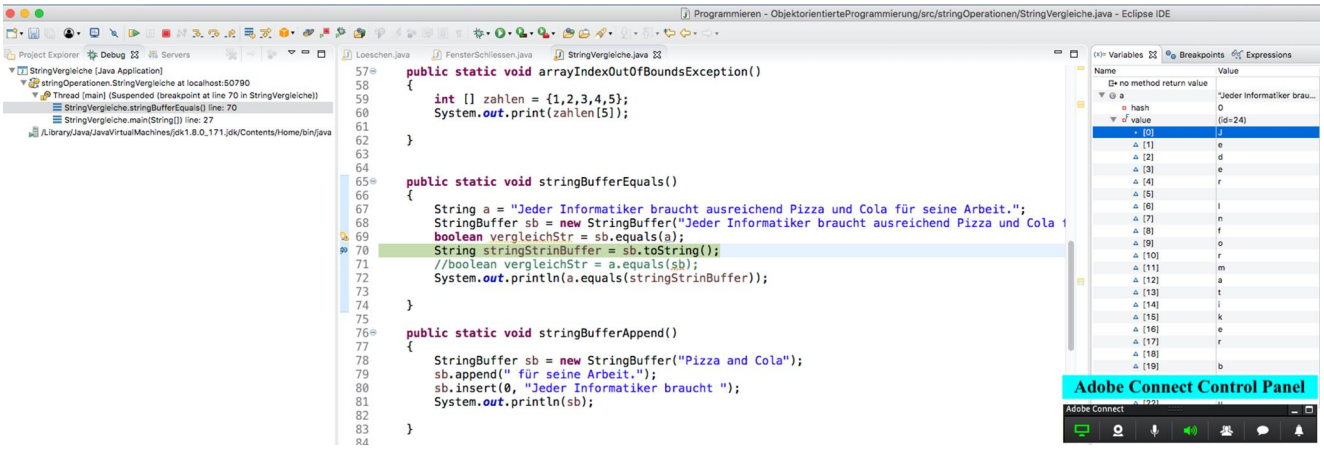


Abb. 5 Debugging in the Eclipse IDE with Adobe Connect (sharing screen pod)

line-Lernplattformen wie code.org oder codeacademy.com zu unseren Vorlesungsinhalten zu ergänzen. Die Lernplattformen ermöglichen den Studierenden direktes Feedback zu den behandelten Themen und den Vorlesungsinhalten. Sie konnte zudem zu vielen Umsetzungsproblemen [28] in OOP weitere Vorschläge erhalten.

Wir empfahlen den Studierenden auch, interne Foren mit ihren Kommilitonen zu nutzen, die nicht regelmäßig genutzt wurden. Wir stellten fest, dass die Studierenden es sich mehr wünschen, sich innerhalb der Vorlesung mit dem Dozenten (wie auch Hölbl et al. für eine MOODLE-Vorlesung [5] herausfanden) oder in den Gruppenräumen auszutauschen.

Schlussfolgerung

Wir fanden heraus, dass im Fernlehren und Fernlernen in OOP Inklusion und Aktivierung die Schlüsselmethoden sind. Wir aktivierten die Studenten, indem wir Gruppenräume einrichteten und Diskussionen mit den Studenten führten oder Fragebögen zur Verfügung stellten. Dadurch blieben die Studenten während der gesamten Vorlesung motiviert. Wir zeigten in einer IDE die gelehrten Programmiermethoden und -konzepte bröckchenweise und lieferten zu abstrakten Erklärungen konkrete OOP-Beispiele. Dies war sehr nützlich, um Frustrationen bei den Studierenden beim Lernen von OOP zu verringern und die Zufriedenheit unserer Studenten zu erhöhen. OOP-Vorlesungen, die in einer virtuellen Lern- und Lehrumgebung abgehalten werden, könnten die hohe Abbruchrate von OOP-Kursen mit der Anwendung einer Vielzahl von Lehr- und Lernmethoden im gleichen Lernkontext verringern, z. B. durch den Einsatz von Adobe Connect mit seinen Pods und Funktionalitäten.

Danksagung

Wir danken der DIPLOMA Fachhochschule Bad Sooden-Allendorf und Prof. Dr. Michael Namokel für die Veröffentlichung dieser Forschungsergebnisse. Wir danken auch Haitam Ben Yahia und Felix Schmidt für das Korrekturlesen der Arbeit.

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

1. Kim K-J, Bonk CJ (2006) The future of online teaching and learning in higher education. Educ Q 29(4):22–30
2. Anderson T (2004) Teaching in an online learning context. Theory Pract Online Learn 273:276–279. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.9849&rep=rep1&type=pdf
3. Yuan L, Powell S (2013) Moocs and open education: implications for higher education
4. Giraffa LM, Moraes MC, Uden L (2014) Teaching object-oriented programming in first-year undergraduate courses supported by virtual classrooms. In: The 2nd international workshop on learning technology for education in cloud. Springer, Berlin Heidelberg, S 15–26

5. Hölbl M, Welzer T (2010) Students' feedback and communication habits using moodle. Elektronika Elektrotechnika 102(6):63–66
6. Khalil H, Ebner M (2014) Moocs completion rates and possible methods to improve retention-a literature review, in EdMedia+ innovate learning. Association for the Advancement of Computing in Education (AACE), S 1305–1313. Konferenz: World Conference on Educational Multimedia, Hypermedia and Telecommunications, Land: Finnland; https://graz.pure.elsevier.com/de/publications/moocs-completion-rates-and-possible-methods-to-improve-retention
7. Lalley J, Miller R (2007) The learning pyramid: does it point teachers in the right direction. Education 128(1):16
8. Kumar A (2009) Personal, academic and career development in higher education: SOARing to success. Routledge, London. https://doi.org/10.4324/9780203938348
9. Race P (2019) The lecturer's toolkit: a practical guide to assessment, learning and teaching. Routledge, London. ISBN 978-0367182267
10. Race P (2014) Making learning happen: A guide for post-compulsory education. SAGA Publications. ISBN 978-1446285961
11. Holland S, Griffiths R, Woodman M (1997) Avoiding object misconceptions. In: Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education, S 131–134
12. Ramsden P (2003) Learning to teach in higher education. Routledge, London. https://doi.org/10.4324/9780203507711
13. Godden DR, Baddeley AD (1975) Context-dependent memory in two natural environments: on land and underwater. Br J Psychol 66(3):325–331
14. Jung M (2015) Fernunterrichtsstatistik 2014 veröffentlicht. https://www.fernstudium-infos.de/fernstudium-rundschau/fernunterrichtsstatistik-2014-ver%C3%B6ffentlicht-r52/. Zugegriffen: 14.05.2020
15. DK, Hemmings J (2018) How psychology works: applied psychology visually explained
16. Forehand M (2010) Bloom's taxonomy. Emerg Perspect Learn Teach Technol 41(4):47–56
17. (2019) U. Academic Computing Services Columbia University, What are pods? https://sites.google.com/a/tc.columbia.edu/adobe-connect-help/what-are-pods. Zugegriffen: 14.05.2020
18. Adobe Inc (2019) Start, attend, and manage adobe connect meetings and sessions. https://helpx.adobe.com/adobe-connect/using/starting-attending-meetings.html. Zugegriffen: 14.05.2020
19. Race P (2019) Updated powerpoint of 'ripples model. https://phil-race.co.uk/2016/04/updated-powerpoint-ripples-model/. Zugegriffen: 14.05.2020
20. Gannod G, Burge J, Helmick M (2008) Using the inverted classroom to teach software engineering. In: 2008 ACM/IEEE 30th International Conference on Software Engineering. IEEE, New York, S 777–786. https://dl.acm.org/doi/10.1145/1368088.1368198
21. Talbert R (2012) Inverted classroom. Colleagues 9(1):7
22. Waldersee R, Luthans F (1994) The impact of positive and corrective feedback on customer service performance. J Organiz Behav 15(1):83–95
23. Maalej W, Tiarks R, Roehm T, Koschke R (2014) On the comprehension of program comprehension. ACM Trans Softw Eng Methodol 23(4):1–37
24. Maalej W, Ellmann M, Robbes R (2017) Using contexts similarity to predict relationships between tasks. J Syst Softw 128:267–284
25. Tulving E (1985) How many memory systems are there? Am Psychol 40(4):385
26. Bandura A (2008) Observational learning, The international encyclopedia of communication
27. Cowan N (2001) The magical number 4 in short-term memory: a reconsideration of mental storage capacity. Behav Brain Sci 24(1):87–114
28. Heeb H (2001) Overcoming the problem of cognitive load in object-oriented programming by microworlds. http://heeb.ch/java/microworlds/oopmmal.pdf

Informatik Spektrum (2021) 44:122–125
https://doi.org/10.1007/s00287-021-01338-y

AKTUELLES SCHLAGWORT

Software Engineering

Ernst Denert[1]

Angenommen: 13. Januar 2021 / Online publiziert: 10. Februar 2021

Der Software-Engineering-Preis wird seit 1992 vergeben, bis 2018 von der Ernst-Denert-Stiftung für Software-Engineering, neuerdings vom GI-Fachbereich Softwaretechnik, finanziert von der Gerlind & Ernst-Denert-Stiftung. Dieser Text ist das Geleitwort für den Band „Ernst Denert Award for Software Engineering 2019“, in dem die Preiskandidaten ihre Arbeiten vorstellen.

1968

„What we need, is software engineering“, sagte F.L. Bauer 1968 und organisierte die Konferenz in Garmisch, die unser Fach begründete. Nachzulesen in seinem Bericht über diese Konferenz im *Informatik-Spektrum* 25 Jahre danach:

> Verärgert darüber, daß am Ende nicht mehr als ein weiteres rein wissenschaftliches Projekt herauskommen könnte, bemerkte ich eines Tages: „The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be.“ Als ich feststellte, daß dies einige meiner Kollegen schockierte, setzte ich mit dem Ausspruch nach: „What we need, is software engineering.“ Das schlug ein. (Zitiert nach [1])

Im Konferenz-Band [2] heißt es so:

> The phrase “software engineering” was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.

Softwareentwicklung sollte also eine Ingenieursdisziplin werden mit systematischer Arbeitsweise, etwa dem Schema folgend

> Planen – Konstruieren – Fertigen – Prüfen – Installieren – Warten.

Wie es einen Bauplan für ein Haus und eine Konstruktionszeichnung für eine Maschine braucht, bedarf ein Softwaresystem einer Architektur.

Softwarearchitektur

Softwarearchitektur ist die Königsdisziplin des Software Engineering. Sie befasst sich mit der Gestaltung von Softwaresystemen im Ganzen, mit ihrer fachlichen Funktionalität und den Schnittstellen zur Außenwelt sowie der inneren Struktur im Großen wie im Kleinen. Bei der Gestaltung im Großen geht es um den Aufbau eines Systems aus Modulen, deren Interaktion in ausführbaren Prozessen und um die Verteilung des Systems auf Hardwarekomponenten. Im Kleinen geht es um die Gestaltung einzelner Module: um ihre Außensicht einerseits, d. h. um ihre Schnittstelle zu anderen Modulen, und um ihre innere Struktur andererseits, also darum, mit welchen Funktionen sie auf welchen Daten operieren.

Das Wort Architektur hat sich in der IT als „gehobene“ Bezeichnung für Strukturen verschiedener Art eingebürgert, so auch bei Software. Softwarearchitektur ist ein viel strapaziertes Schlagwort mit vielerlei Interpretationen, manchmal auch mit dem Verständnis, Architektur an sich sei etwas Gutes. Das stimmt nicht, es gibt auch schlechte Architektur. Manchmal heißt es: „Unsere Software hat keine Architektur“. Auch das stimmt nicht, jedes System hat eine. Meist ist gemeint, dass seine Struktur schlecht oder uneinheitlich ist, undokumentiert nur in den Köpfen einiger Programmierer steckt oder gänzlich unbekannt ist. Sie existiert dennoch.

✉ Ernst Denert
ernst.denert@web.de

[1] Grünwald, Deutschland

Solche Strukturen werden häufig grafisch mit Kästchen und Strichen dazwischen dargestellt, meist ohne präzise festzulegen, was sie bedeuten. Die Kästchen stehen für Komponenten, Module, Funktionen, Dateien, Prozesse, die Striche und Pfeile für Beziehungen aller Art, Verweise, Datenflüsse, Aufrufe und manches mehr. Oft herrscht darüber keine Einigkeit, nicht einmal innerhalb eines Projekts, einer Abteilung, schon gar nicht innerhalb von ganzen Unternehmen der Wirtschaft, ebenso wenig in wissenschaftlichen Instituten.

Als *Softwarearchitektur* bezeichne ich die statischen und dynamischen Strukturen, die ein Softwaresystem prägen, betrachtet aus 3 Perspektiven, *Sichten* genannt: Anwendungs-, Konstruktions- und Programmsicht. In jeder wird anders über das System gedacht. Denken braucht Sprache. Alle Menschen denken in ihrer Muttersprache, Musiker in Noten, Architekten in Zeichnungen, Mediziner in lateinischen Fachbegriffen, Mathematiker in Formeln. Und Informatiker denken in Programmiersprachen. Das genügt aber nicht, um Softwarearchitektur zu gestalten, jede Sicht hat ihre eigenen Begriffe, Konzepte und Darstellungsformen:

1. *Anwendungssicht*
   Dafür muss man die Sprache der Anwender verstehen und in Konzepten und Begriffen der Anwendung denken, welcher Art sie auch sein mag, betriebswirtschaftlich, technisch, wissenschaftlich, medial etc. Bei *betrieblichen* Systemen geht es um Daten, fachliche Funktionalität und Geschäftsvorgänge sowie Interaktion mit Benutzern (Dialoge) und Nachbarsystemen, bei *technischen* Systemen um die physische Maschinerie und Regelungsverfahren. Fachliche Sachverhalte werden mit natürlicher Sprache und mehr oder weniger formalen Darstellungen beschrieben.
2. *Konstruktionssicht*
   Hier geht es um die modulare Gestaltung eines Softwaresystems, d. h. seinen Aufbau aus Bausteinen mit ihren statischen und dynamischen Beziehungen. Grafische Darstellungen spielen eine große Rolle, informelle ebenso wie solche mit formaler Syntax und Semantik, etwa UML. Zudem wird auch hier natürliche Sprache verwendet. Einerseits prägen die fachlichen Konzepte der Anwendungssicht die Konstruktion, sie sollen darin deutlich erkennbar sein. Und zum anderen wird in der Konstruktion die Systemplattform festgelegt, auf der die Software läuft.
3. *Programmsicht*
   Hier dominieren formale Sprachen: Programmiersprachen, Schnittstellen von Frameworks und Bibliotheken (APIs), Kommunikationsprotokolle. Entwicklungsumgebung und -werkzeuge sowie Programmierregeln prägen die Arbeitsweise. Die Programmstrukturen im Großen sind durch die Konstruktion vorgegeben, insbesondere die Verzeichnisstruktur des Quellcodes durch die Modularisierung.

Auf den ersten Blick erscheinen die Architektursichten wie *Entwicklungsphasen*. Man kann sie auch so sehen, ich jedoch betrachte die Architektur eines Softwaresystems in erster Linie vom Ergebnis her – also wie sie ist oder sein soll(te) –, weniger in ihrer Entstehung. Man stelle sich vor, ein betriebsbereites Softwaresystem müsse vor seinem Einsatz abgenommen und zugelassen werden, wie etwa Züge, neue Automodelle oder medizinische Geräte. In der dazu nötigen Prüfung wird seine *tatsächliche* Architektur untersucht, gestützt auf ihre Dokumentation und vor allem wie sie sich im Code manifestiert, und nicht wie sie entstanden ist. Bei dieser ergebnisorientierten Betrachtung spielen Entwicklungsprozesse und Vorgehensmodelle keine Rolle, gleichgültig ob phasenorientiert oder agil.

Immer wieder begegnet man der Unterscheidung zwischen „grob“ und „fein“, etwa als Grob- und Feinspezifikation. Das ist sinnvoll, wenn die grobe Form eine präzise Abstraktion der feinen ist, nicht jedoch, wenn grob nur bedeutet „so ungefähr“. Eine Grobdarstellung ist auch gut, wenn sie einen guten Überblick gibt über einen komplexen Sachverhalt mit vielen Details. Keinesfalls jedoch darf man Anwendungssicht als grob verstehen, die Konstruktionssicht als ihre Verfeinerung. Präzision auch im Detail ist in allen Sichten wichtig.

Softwareentwicklung

Ingenieurgemäße Softwareentwicklung basiert natürlich auf einem Architekturplan, allerdings gibt es in der Praxis auch heute noch viel „tinkering“ (F.L. Bauer), nur wird es nicht mehr basteln genannt, sondern agil.

Über die Jahrzehnte sind viele Konzepte für das Vorgehen in Softwareprojekten publiziert und praktiziert worden. Für das strukturierte Vorgehen in Phasen wurden diverse Bezeichnungen anhand grafischer Darstellungen geprägt: Wasserfall-, Spiral-, V-Modell. Diese Metaphern sind sachlich wenig erhellend, aber Gegenstand weltanschaulicher Auseinandersetzungen.

In großen Organisationen, insbesondere auch in staatlichen, gibt es eine Tendenz zu formal-bürokratischer Ausformung von Phasen- und Reifegradmodellen. Die bekanntesten Beispiele sind das Capability Maturity Model (CMMI) des SEI in den USA und das deutsche V-Modell. Auf der anderen Seite und als Gegenbewegung sind vor gut 20 Jahren die agilen Methoden entstanden, beginnend mit Extreme Programming (XP), heute überwiegend Scrum.

Den Agilen ist es fremd, einen Plan zu machen; Anforderungen und einen Architekturentwurf aufzuschreiben,

gilt als vertane Zeit und Mühe. Stattdessen wird in einer Folge von Schritten, meist Sprints genannt, gleich etwas programmiert, das dem Anwender taugen könnte. Wenn dabei verkorkste Programmstrukturen entstehen, wird Refactoring gemacht. Das ist doch seltsam: Man macht sich erst mal keine Gedanken über die Softwarestruktur, aber wenn sie dann nach mehreren Programmiersprints verdorben ist, fängt man an zu reparieren. Das ist kein strukturiertes Arbeiten. Den agilen Methoden mangelt es an ingenieurgemäßer Methodik; konsequenterweise sagt der Scrum-Guide nichts zu Software Engineering.

Bertrand Meyer bietet in seinem Buch „Agile!“ eine profunde, sachliche Darstellung, Analyse und Beurteilung der agilen Methoden. Dem habe ich nur eines hinzuzufügen: Das Beste an den agilen Methoden ist ihre Bezeichnung: agil – ein grandioser Marketing-Gag. Wer traut sich denn zu sagen: Wir arbeiten nicht agil, sondern strukturiert?

Auf den Punkt gebracht: Egal wie ein Team arbeitet, mit Ingenieurmethode, agil oder sonst wie – *die Softwarearchitektur muss bedacht werden*. Man kann sie vor dem Programmieren entwerfen und auch aufschreiben oder genialisch in den Code hacken – *man muss die Softwarearchitektur bedacht haben*. Am Ende ist auf jeden Fall etwas niedergeschrieben – im Code. Hoffentlich taugt der. *In code veritas.*

Teamarbeit

Softwaresysteme sind komplex und groß, sie können aus Hunderttausenden, gar Millionen Zeilen Code bestehen. Ein Einzelner kann sie nicht entwickeln, Teamarbeit ist nötig. Diese setzt Struktur voraus: in der Software eine Architektur und im Team eine Aufgabenteilung; nicht jeder kann alles jederzeit machen.

In jüngerer Zeit wird Teamarbeit verklärt: Bunt gemischte Gruppen, ohne Chef, in flachen Hierarchien, eigenverantwortlich und selbstorganisiert, kriegen alles locker hin in täglichen Stand-up-Meetings, von Sprint zu Sprint. Tiefes und gründliches Nachdenken Einzelner, beispielsweise über Architekturaspekte wie die Modularität einer Software, ist nicht gefragt. Im Wikipedia-Artikel über Scrum steht: „Das Entwicklungsteam ist für die Lieferung ... verantwortlich ... organisiert sich selbst“.

Verantwortung kann man nicht einem Team zuschreiben, sondern nur einzelnen Personen. Und ein professionelles Projektteam bildet sich nicht wie eine Gruppe von Freunden zur Organisation einer Party, es wird erst einmal zusammengestellt von jemandem mit Personalverantwortung. Wenn dann nichts weiter festgelegt wird, bildet sich zwar eine informelle Organisation und auch eine Führung, aber es mangelt an klaren, verbindlichen Zuständigkeiten. Ein Freund karikiert das mit dem Spruch: „Team = **T**oll, **e**in **a**nderer **m**acht’s“.

Eine meiner fundamentalen (Berufs-)Lebenserfahrungen lautet dagegen: Menschen wollen und müssen geführt werden. Selbstverständlich kommt die militärische Art der Führung mit Befehl und Gehorsam für Softwareentwickler nicht infrage, eher schon die des Fußballtrainers einer Profimannschaft oder des Dirigenten eines Sinfonieorchesters. Ein Projektleiter muss fachlich kompetent und überzeugend sein, zugleich kommunikativ und motivierend, die Mitarbeiter anleiten und dafür sorgen, dass sie gute Arbeitsergebnisse erzielen (können). Anlässlich des 25. Jahrestages der Garmischer Software-Engineering-Konferenz habe ich in einem Artikel im Informatik-Spektrum etwas geschrieben, das auch heute noch gilt:

> Ein guter Geist im Team ist für den Erfolg eines Softwareprojekts wichtiger als alle Technik. Deshalb muss sich das Management ständig um die Bedingungen für ein gutes Klima bemühen – ein Manager muss Berufslebensqualität schaffen. Entstehen kann das Klima natürlich nur im Team selbst, alle tragen dazu bei; je besser Verständigung und Verständnis, desto besser das Klima. [3]

Zum Schluss ein Wunsch

Ein Rückblick auf ein Vierteljahrhundert Software-Engineering-Preis zeigt mir, dass sich ein Großteil der Arbeiten mit analytischen Verfahren befasst, etwa Analyse von Code und Modellen, konstruktive Methoden kommen dagegen kaum vor. Bedauerlich, denn Software Engineering ist die Lehre vom Gestalten, Konstruieren, Bauen und Entwickeln von Softwaresystemen. Warum gibt es nicht an einigen Universitäten eine Software-Engineering-Schule, die normativ sagt: So macht man es? Es dürfen gerne zwei oder drei miteinander wetteifern, eventuell mit unterschiedlicher Ausrichtung hinsichtlich der Systemarten, etwa technische, betriebliche, mediale Systeme.

Liegt es an der Praxisferne oder am kurzatmigen Wissenschaftsbetrieb, in dem vor allem sechs seitige Artikel publiziert werden mit dem Ziel, den Zitierungsindex zu verbessern? Leider werden kaum noch Lehrbücher geschrieben. Ich wünsche mir eine konstruktive Software-Engineering-Lehre, als Buch, gerne auf Papier und natürlich digital, ergänzt durch eine Website mit Code-Beispielen und weiteren Materialien.

Die meisten Informatik-Absolventen, die in die Praxis der Wirtschaft gehen, arbeiten an der Entwicklung von Software, für neue Systeme ebenso wie für bereits laufende. Ihnen auf den Weg mitzugeben, wie man das konkret macht, dem sollte eine solche Lehre dienen. Vielleicht heißt es im

Unternehmen dann mal: Wir folgen der X-Schule – das wär' doch was.

Und es könnte Software-Engineering, unserem praktisch sehr relevanten Fach, wieder zu Ansehen in der Öffentlichkeit verhelfen, aus der es, ebenso wie die Informatik, verschwunden ist, überwuchert von den allgegenwärtigen Schlagwörtern.

Literatur

1. Bauer FL (1993) Software Engineering – wie es begann. Informatik Spektrum 16(5):259
2. SOFTWARE ENGINEERING, Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 7.–11. Okt. 1968, Kap. 1, S. 13.
3. Denert E (1993) Software-Engineering in Wissenschaft und Wirtschaft: Wie breit ist die Kluft? Informatik Spektrum 16(5):299

Informatik Spektrum (2021) 44:126–128
https://doi.org/10.1007/s00287-020-01326-8

KOLUMNE

Digitaler Workaround ist keine Digitalisierung

Wem berechtigte Kritik egal ist, der siecht dahin

Gunter Dueck[1]

Angenommen: 18. November 2020 / Online publiziert: 21. Dezember 2020

Zusammenfassung
Immer noch wird „Digitalisierung“ wie eine unangenehme Anpassung an die jeweils neue Lage empfunden. Die fast erzwungenen Maßnahmen wirken wie eine neue Oberfläche, unter der eigentlich alles bei Alten bleibt. Echten Problemen wird ausgewichen, man versucht es mit Workaround. Statt einer digitalen Neuerfindung migriert man auf neuere Technologien, immer wieder – was nur Geld kostet, aber im Sinne des Business nichts bringt. Es geht kein Ruck durch die IT und schon gar nicht durch die Vorstandsetagen. Viele Unternehmen zucken nur nch mit den Achseln, wenn Kunden eine moderne IT anmahnen. Eine solche Haltung könnte man „Einigeln“ nennen – das ist die Unternehmensphase vor dem Tod. Was muss geschehen? Ausigeln, ganz klar.

Seit langer Zeit wird über Disruptionen geredet und geschrieben. Das initiale Buch dazu, *Innovator's Dilemma* von Clayton Christensen, erschien im vorigen Jahrhundert. Ich schreibe diesen Artikel mit Office 365, das heute Morgen aktualisiert wurde. Das Wort Disruption wird immer noch als Rechtschreibfehler angekreidet. Eine Disruption hat oft zur Folge, dass Unternehmen durch etwas Neues total auf dem falschen Fuß erwischt werden. Das ist sehr unangenehm bis desaströs. Die derzeitige Pandemie verstärkt den Disruptionsdruck sehr.

Schauen wir eine solche Lage genauer an:

- Viele Mitarbeiter sind ganz überflüssig geworden; viele, manchmal sogar die meisten, haben eine falsche Qualifikation – wohlgemerkt, eine falsche, nicht bloß eine zu niedrige (das wäre noch zu heilen).
- Die Produkte oder Services sind „out“.
- Die Produktionsmethoden sind überholt und zu teuer.
- Die gewohnten Prozesse und Denkweisen passen nicht zum Neuen.
- Die Controller haben die Prozesse im Alten optimiert, sie haben aber keinerlei Erfahrung oder Qualifikation für die Schaffung neuen Prozesse (das ist ein noch ganz undiskutiertes Thema; weil das so ist, bildet sich hier unerkannt der Kern von Großkatastrophen).
- Wegen falscher Prozesse hat das Unternehmen eine falsche Betriebskultur.
- Das Management behandelt Wissensarbeiter wie unmündige Prozesssklaven.
- Die Gehaltsstrukturen sind outdated.
- Das Unternehmen als Ganzes weigert sich, große Veränderungen zu stemmen, weil das Gewinnwachstum damit unterbrochen und eine Gewinnprognose schwierig werden würde.
- Zu dem Hype „New Work“ muss sich auch der des „New Management“ gesellen, viele traditionell verdiente Manager müssen ausgewechselt werden, mindestens werden Machtverschiebungen zwischen den Bereichen notwendig.
- Besonders große Unternehmen willigen unter dem Druck der Betriebsräte ein, interne Verlierer einer Veränderung zu entschädigen; Autoproduzenten schließen zum Beispiel Kündigungen auf erstaunlich lange Frist aus, obwohl klar ist, dass die unter Schutz Gestellten in wenigen Jahren eine falsche Qualifikation haben werden. Das muss nicht, kann aber eine Katastrophe einleiten.

Vergleichen Sie Zweigstellenbanken mit Internetbanken und gehen sie dabei die obigen Punkte durch. IBM hat lange „IT aus der Steckdose“ prognostiziert. Die kommt jetzt, aber diese IT verkauft sich quasi per Selbstbedienung

✉ Gunter Dueck
gunter.dueck@googlemail.com

[1] Neckargemünd, Deutschland

und braucht keinen hochprofessionellen Vertrieb, der in der Hierarchie der IBM in der Schlüsselstellung ist („Sie können hier nur Karriere machen, wenn Sie etwas im Vertrieb gerissen haben"). Autos bestehen aus immer komplexeren Teilen, die per Software digital verbunden werden, nicht mehr mechanisch, nicht mehr elektronisch. Bei den vielen Sensoren und „embedded devices" nähert man sich einem Komplexitätskollaps. Tesla hat es anders gemacht. Dort entwickelt man erst die Software („Betriebssystem") und baut nur solche Devices ins Auto ein, die „plug & play" dazu passen (damit ist einst Microsoft aufgestiegen; Windows 95). Diesen Sachverhalt hat letztens der AUDI-Chef Markus Duesmann in einem SZ-Interview erklärt, er sieht sich vor harte Probleme gestellt (ich betone, dass es Herr Duesmann gesagt hat, weil mich gleich wieder Tesla-Hasser abkanzeln, die diese Problematik nicht kennen und immer auf den schlechten Spaltmaßen von Tesla herumhacken). Tesla verkauft nur im Internet, das geht bei anderen Autos nicht, weil man im Autohaus Stunden und Stunden mit Feilschen verschwendet und dann mit 15 bis 20 % Ersparnis herauskommt, was zusätzlich eben noch Prachtbauten, Probefahrten und Verhandeln kostet. Wer im Internet verkauft, muss wie Tchibo oder Apple einen einzigen unumgänglichen Preis festsetzen.

Weiter: Home-Office leert Bürotürme, es könnte eine Immobilienblase platzen lassen. Wenn eine Firma viel Home-Office anbietet, werden Einzelbüros zu teuer, etc. Oder: Lesen Sie unter „Stellwerk" in Wikipedia diesen Satz: „Ende 2013 verfügte die Deutsche Bahn nach eigenen Angaben über 3397 Stellwerke, die im Durchschnitt 47 Jahre alt waren."

Gehen Sie in allen diesen Fällen die obigen Punkte durch; Sie sehen, dass es nicht einfach nur um „Digitalisierung" geht. Nein, es geht in der Mehrheit der Fälle dem ganzen System an den Kragen.

Das aber wehrt sich, indem es sich zäh mit allerlei Workarounds behilft. Die ersten E-Books wurden als PDF zum Ausdrucken ausgeliefert – dabei könnte man reine E-Books viel schöner designen, weil Farbabbildungen und Links nichts kosten. Das E-Book ist auch heute noch nur der Abklatsch der alten Printausgabe, die die Form diktiert. Ebenso „digitalisiert" man das Grundbuch erst einmal wie ein PDF, damit man es exakt gleich auf dem Bildschirm sehen kann, und die Mitarbeiter werden bestimmt befördert, weil sie jetzt zusätzlich den Bildschirm-Digital-Skill beherrschen müssen (ist Spaß, aber ich bekomme in diesem Stil öfter erschütternde Leserbriefe aus Verwaltungen). Die Schulen schicken jetzt die Hausaufgaben per Mail, sie verteilen Tablets, aber richtiger Unterricht geht noch nicht digital, obwohl hier seit Jahrzehnten E-Learning thematisiert wird und in anderen Ländern schon gut klappt.

Das überall so sehr zögerliche Befassen mit dem Digitalen, die Lippenbekenntnisse und Workarounds wirken wie hinhaltender Widerstand. Man hat im Prinzip nichts gegen die Digitalisierung, aber sie soll keine Schmerzen bereiten, insbesondere müssen Machtstrukturen und Meetingkaskaden bleiben und außerdem soll lieber niemand umgeschult werden müssen. „Wir liefern dem Kunden nur das, was wir gut können. Der Kunde muss daher verstehen lernen, was wir können und was eben nicht." Bei vielen Konzernen ist es wie damals in einem Ostblock-Restaurant. „Haben Sie Wiener Schnitzel?" – „Nein." – „Putenröllchen?" – „Nein." – „Pommes Frites?" – „Nein." – „Oh, vielleicht geht es anders herum schneller: Was haben Sie denn alles?" – „Mixed Grill."

„Kann die Bank nicht öffnen, wenn ich Zeit habe?" – „Nein." – „Kann die Bahn nicht pünktlich kommen?" – „Nein." – „Kann ich meinen Rentenantrag im Internet stellen?" – „Das könnte technisch klappen, macht aber keinen Sinn, weil Sie das ohne Beratung sowieso nicht verstehen." Ich habe damals geschluckt. Ich habe zaghaft vorgebracht, ich wäre Akademiker, das hat aber nicht überzeugt ...

Ich erkläre es jetzt einmal andersherum und komme zum Punkt: Die Protagonisten von Innovationen stoßen meist auf Gelächter und Unverstand. Erst wenn sie es so weit bringen, dass „Early Adopter" ihre Prototypen interessiert ausprobieren, erst, wenn die Preise stimmen und eine erste aufgeschlossene Käuferschicht das gebotene Neue gerne nutzt, dann kommt die Innovation groß heraus. Sie geht einen langen Spießrutenlauf: „Das geht noch nicht, das muss anders, hier muss ein neues Konzept her ..." Ein guter Ratschlag ist, bei jeder Kritik genau hinzuhören, ob sie einen wichtigen Punkt aus vernünftiger Sicht trifft. Wenn das so ist: Kundenwunsch erfüllen! Sofort! Ohne Murren! Ich weiß, das ist schwer zu schlucken, deshalb bringen es uns Berater als „design-gethoughtetes agiles rapid prototyping" bei, egal: Jede vernünftige Kritik muss weg.

So entstehen die erfolgreichen Innovationen. So entzücken sie die Welt und durchdringen sie. Die Innovationen fassen Fuß, wenn die Bedenken gegen sie langsam ausgeräumt sind. Bei den Banken, Autoproduzenten, Kohlestromlieferanten, bei so ziemlichen allen Verwaltungen und Managemententscheidungskaskaden, den ätzenden Prozeduren der Politik und so ziemlich bei allem Alten wächst dagegen die Kritik: „Warum geht das nicht online, nicht schneller, nicht pünktlicher, nicht klimafreundlicher, gesünder, freundlicher, nicht bargeldlos, nicht digital?"

Da schüttelt sich das Alte leider, leider nicht wirklich. Es ist seiner selbst so sicher, dass es die Kritik eher unbedarft empfindet. „Ihre Kritik zeigt uns, dass Sie nicht tief in der Materie stecken. Sie verstehen die Komplexität nicht. Ihre Kritik macht es sich zu einfach."

„Warum ist die Bahn nicht pünktlich?" Augenrollen. „Warum werden schlechte Lehrer nicht gefeuert oder umgeschult für etwas, das sie können?" Augenrollen. „Warum besteht der Staat darauf, IT-Experten lausig zu bezahlen?"

Augenrollen. „Warum lässt man auch schlechte Professoren Online-Vorlesungen halten – man könnte doch die besseren Vorlesungen aus anderen Unis zuschalten oder gleich alles vom Großlehrmeisterguru dieses Fachs erfahren?“ – „Was sollen wir denn mit den ganzen Professoren machen? Ich meine, so digital muss es nun wieder nicht sein.“

Innovationen beginnen zu glänzen, wenn sie die anfänglichen Kritiken in Begeisterung transformieren. Das ist oft beschwerlich und mühevoll. Dann leben sie auf und erstrahlen.

Das Alte aber wird immer mehr kritisiert. Es zuckt leider, zum Tode geweiht, mit den Achseln, versucht digitale Workarounds, wofür es nicht gelobt wird, und regiert auch deshalb zunehmend verärgert. Wenn das Alte aber zu sehr, zu oft und später zu empört kritisiert wird, dreht es den Spieß um und entgegnet mit Zorn so: „Zum Kotzen! Immer nur Lehrer-Bashing, Bahn-Bashing, BWLer-Bashing, Politiker-Bashing.“ Diese Replik bedeutet, dass wir es hinnehmen sollen und müssen, wie es ist. Das ist der Vorbote des Todes des Alten. „Komm, lass den Opa, du änderst ihn nicht mehr.“

Wir müssen also nicht nur immer das Entstehen des Neuen diskutieren, sondern auch das Siechen des Alten verstehen. Alt ist, was sich auf berechtigte Kritik nicht mehr ändert. Darwin sagt, dass ... – ach, ich lasse das einmal, das wissen Sie ja.

Lernen Sie einfach nie, mit berechtigter Kritik zu leben.

Informatik Spektrum (2021) 44:129–130
https://doi.org/10.1007/s00287-021-01346-y

REZENSION

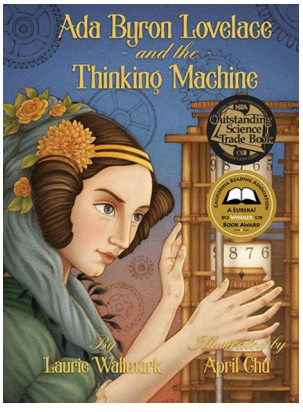
Buchrezension

Ada Byron Lovelace and the Thinking Machine

Frank J. Furrer[1]

Online publiziert: 17. Februar 2021

Laurie Wallmark (Autor), April Chu (Illustrator) Ada Byron Lovelace and the Thinking Machine
Creston Books, Berkeley, California, USA, 2015
ISBN 978-1-939547200



Dieser Lockdown ist für uns Informatiker eine harte Zeit: kein persönlicher Kontakt mit Kollegen und Studenten, ZOOM's ohne Ende und ausgeschlossen vom Campus.

Was könnte man da zum Lesen empfehlen – lehrreich, unterhaltend und wohltuend? Glücklicherweise gibt es eine schöne Antwort: „Ada Byron Lovelace and the Thinking Machine" – ein wunderschöner, instruktiver, und liebevoll gestalteter Bildband.

Auf 20 eindrücklichen, sorgfältig gezeichneten Blättern wird das Leben von Ada Byron Lovelace (1815–1852, Abb. 1) erzählt:

- Ihre schwierige Jugend mit ihrem Vater Lord Byron (berühmter Poet),
- die Flucht mit ihrer Mutter nach London,
- ihr frühes Interesse an Zahlen und Mathematik,
- ihre ersten Erfindungsversuche der „Flying Machine",
- die Aerodynamikversuche mit dem Modellsegelschiff,
- ihre schlimme Masernerkrankung,
- ihre unvollständige Genesung und ihre Privatlehrerin Mary Fairfax Somerville,
- das erste Zusammenreffen mit Charles Babbage,
- die Arbeit mit Charles Babbage an der „Difference Engine",
- der Entwurf der „Analytical Engine",
- ... und dann der historisch erste Entwurf für eine programmierbare Maschine – das erste Programm!
- Mehr als 100 Jahre vor der Erfindung des vollprogrammierbaren Rechners hatte sie den Beruf des Programmierers erfunden (was allerdings von gewissen Historikern angezweifelt wird).

1977 wurde die vom US Department of Defense entwickelte neue Programmiersprache als „ADA" nach ihr benannt. ADA ist eine stark formalisierte Programmiersprache, die heute speziell in sicherheitskritischen Anwendungen eingesetzt wird.

Nach dieser vergnüglichen Lektüre kann man tiefer in das faszinierende Thema eintauchen mit:

- Leben und Werk von Ada Lovelace Byron: Ada Lovelace Byron – *The Making of a Computer Scientist* (von Cristopher Hollings, Ursula Martin, Adrian Rice. University of Chicago Press, Chicago, USA, 2018. ISBN 978-1-851-24488-1)
- Biographie und die Computer-Konstruktionen von Charles Babbage: Charles Babbage from the Beginning (von Lucy May Simister. UK, 2015. ISBN 978-1-51214919-7)

✉ Frank J. Furrer
frank.j.furrer@bluewin.ch

[1] Fakultät für Informatik, Technische Universität Dresden, Dresden, Deutschland

Abb. 1 Charles Babbage, die Difference-Engine und Ada Byron Lovelace (1815-1852) mit freundl. Genehmigung von Creston Books

- Oder sogar in einen Überblick der Programmiersprache ADA: Building Parallel, Embedded, and Real-Time Applications with ADA (von John W. McCormick, Frank Singhoff, Jérôme Hugues. Cambridge University Press, Cambridge, UK, 2011. ISBN 978-0-521-19716-8)

Die Abbildung aus dem Bildband wurde freundlicherweise von Creston Books zur Verfügung gestellt.

Frank J. Furrer, Januar 2021

Informatik Spektrum (2021) 44:131–133
https://doi.org/10.1007/s00287-021-01347-x

FORUM

Gewissensbits – wie würden Sie urteilen?

Stefan Ullrich[1] · **Rainer Rehak**[1]

Online publiziert: 26. Februar 2021

Mit den ethischen Leitlinien der GI haben wir es uns zur Aufgabe gemacht, Diskurse zu ethischen Problemen der Informatik zu initiieren und zu fördern. Mitglieder der Fachgruppe „Informatik und Ethik“ der GI stellen jeweils ein hypothetisches, aber realistisches Fallbeispiel vor, das zur Diskussion anregen soll. Die Fälle können jeweils von Interessierten im Blog der Fachgruppe auf der GI-Website https://gewissensbits.gi.de kommentiert und diskutiert werden.

Fallbeispiel: Smoke & Mirrors

„Geht es dir um die Sache oder um dich?“ Hannah weiß nicht, was sie ihrer Mitbewohnerin antworten soll. „Wenn es dir um die Sache geht“, fährt Jelena fort, „solltest du zusagen. Ethik und Nachhaltigkeit sind doch genau dein Ding. Und ein bisschen mehr Publikum kann doch auch nicht schaden!“ Hannah schaut nochmal auf das Schreiben der großen Organisation „AI & I4 Future“, die auf Social Media nicht zuletzt durch ihre visuell beeindruckenden Kampagnen bekannt wurde. AI&I4F bezeichnet sich selbst als ein visionäres Zukunftslabor und Crash-Test-Institut für Künstliche-Intelligenz-Systeme. In dem Schreiben wird sie eingeladen, zusammen mit Expert*innen und Aktivist*innen aus ganz Europa ein Manifest für eine nachhaltige und gemeinwohlorientierte Künstliche Intelligenz zu erarbeiten. Die Einladungsliste liest sich erstklassig, es sind ebenso gestandene Wissenschaftler*innen darunter wie populäre Schauspieler*innen; zu Veteran*innen der Anti-Atomkraft-Bewegung gesellen sich junge Influencer*innen aus der Ökotech-Szene – kurzum, es scheinen alle relevanten Stakeholder ausgewählt zu sein. „Stakeholder“, wiederholt Hannah plötzlich laut zu sich selbst, „ich denke schon genau so wie SIE. Was soll denn Multi-Stakeholder überhaupt heißen?“ Jelena stutzt und zupft ein wenig verträumt an den zahlreichen Schichten ihrer selbstgenähten Kleidung. „Ist es ein Problem für dich, dass diese AI-Leute einfach so stylisch sind und echt gute Visuals produzieren?“ Hannah will es ja nicht zugeben, aber neben inhaltlichen Fragen ist auch das irgendwie ihr Problem. Sie kommt sich etwas albern vor, doch nach einigem Hin und Her entscheidet sie sich, im Expert*innen-Gremium mitzumachen.

Das erste Treffen aller Beteiligten findet wenige Wochen später auf einem alten Gutshof im Süden des Landes statt, dem Hauptsitz von AI&I4F. Sie werden vom Bahnhof mit Elektroautos abgeholt, Hannah sitzt mit dem bekannten Transformationsforscher Mats Hillebork auf der Rückbank. Durch Plexiglas-Abtrennung und Maske kann sie ihn nicht so gut verstehen, sie ist auch kein *native speaker*, aber er scherzt ganz offensichtlich über den Aufwand des Treffens. Auf Englisch teilt er ihr mit, dass der Gutshof einst der Fugger-Handelsfamilie gehört hat und von AI&I4F schließlich zu einem Vorzeigehof in Sachen Nachhaltigkeit umgebaut wurde. Ob Hannah sich nie gefragt habe, warum eine historische Mühle das Logo von AI&I4f ziere? Aber so richtig ökologisch nachhaltig sei das nicht, meint Hillebork, als sie an der Mühle vorbeisteuern. „Das ist eine Mühle der

✉ Stefan Ullrich
stefan.ullrich@gi.de

[1] Weizenbaum-Institut für die vernetzte Gesellschaft, Berlin, Deutschland

Habsburger aus Spanien, die wurde vor 10 Jahre Ziegel für Ziegel und Balken für Balken hierher geschafft, die alte Bockwindmühle war wohl nicht schick genug!“ Hillebork lacht, und so weiß Hannah nicht, ob es ein kruder Scherz war oder sie auch alles richtig verstanden hat.

Für eine auf Künstliche-Intelligenz-Audits spezialisierte Organisation findet sich erstaunlich wenig Technisches hier. Sicher, die E-Autos sind da, auch die Überwachungskameras, aber ansonsten tummeln sich eine Unmenge von Angestellten auf dem weiten Gelände. Frauen, die Wäsche auf Waschbrettern waschen, junge Männer, die Wasser aus dem Brunnen schöpfen und ein alter Butler, der in der Tür steht und ihnen die Richtung ins Haus weist. Überall stehen Schalen mit Obst und vegetarischen Leckereien. Es ist angenehm kühl im Schloss, ja, es wirkt wirklich wie ein Schloss. Sie hat zwei Zimmer nebst modernem Bad für sich allein, hier sieht alles ultramodern aus, im Gegensatz zum restlichen Gebäude. Die aktuellen Wetterdaten und neuesten Community-Nachrichten erscheinen in einer Ecke des Badezimmerspiegels. Das Kick-Off-Meeting wird drei Tage dauern, macht sie sich klar. „Ich sage schon wieder Kick-Off, ich übernehme schon wieder deren Sprache, das ist nicht Aye Aye, AI“. Die letzten Worte sind der Werbeslogan einer bekannten KI-Sprachassistenz, und tatsächlich: Im Bad ist das charakteristische Doppelklingeln zu hören und ein System wartet auf die Eingabe der überraschten Nutzerin.

Hannah macht sich bereit für das erste Treffen vor dem angekündigten „AI-Dinner“. Sie erwartet ein Arbeitsessen, zieht sich also nicht besonders schick an. Sie ist daher sehr überrascht, als sie unten in der Lobby ankommt und dort viele Fernsehkameras laufen. Es werden Interviews und Fotos gemacht und der ein oder die andere Journalistin geht herum und lässt sich von den Expert*innen die besonderen Möglichkeiten der KI erklären und wie wichtig ein ethischer Umgang damit sei, gerade in Abgrenzung zu den USA und China, wie es heißt. Wie schön wird die Welt der Zukunft, wenn die KI uns so viele Probleme abnimmt und noch dazu exportiert werden kann. Auch die restlichen Aktivitäten der nächsten Tage auf dem Schloss gehen inhaltlich nicht besonders tief, werden von der Moderation oft hin zu Allgemeinplätzen bewegt und sind regelmäßig von öffentlichkeitsorientierten Pausen mit Fototerminen und Interviews durchzogen, teilweise sogar mit internationalen Medien. Hannah ist ein wenig enttäuscht, da sie viel mehr inhaltliche Arbeit erwartet hätte. Andererseits trifft sie viele interessante Menschen aus ihrem Themenbereich, und auch die paar Tage mal ganz aus ihrem Alltag zu sein genießt sie sehr, wie sie sich eingesteht. Vielleicht muss sowas auch ab und zu einfach mal sein und sie sollte das alles nicht zu eng sehen, verdient hat sie es jedenfalls durch ihre harte Arbeit den Rest der Zeit.

Diese Art Treffen finden nun alle paar Monate statt und innerhalb von zwei Jahren entwerfen die Expert*innen ein Abschlussdokument, unterzeichnet von allen Beteiligten, optisch sehr ansprechend gestaltet und prominent durch die aktuelle AI&I4F-Kampagne beworben, die ein optionales KI-Güteprüfsiegel für europäische Value-Based-AI propagiert. Inhaltlich ist das Dokument jedoch Hannahs Ansicht nach wenig visionär, gibt im Kern nur den allgemeinen Stand der Diskussion wieder und bleibt auch sonst weit hinter den Ideen, Vorstellungen und Möglichkeiten der Beteiligten zurück. Und auch von der Kampagne war anfangs keine Rede gewesen, deren Ziele sie so gar nicht teilt, weil sie im Prinzip eine „freiwillige Selbstkontrolle der Wirtschaft“ bewirbt. Hannah ärgert sich letztendlich doch, dass sie so viel Zeit und Energie hineingesteckt hat, wenn am Ende sowas dabei herauskommt, was sie am Anfang ja eigentlich schon kritisiert und befürchtet hatte: Das Papier enthält viele oberflächliche Allgemeinaussagen – denn wer wäre denn gegen „vertrauenswürdige KI im Sinne des nachhaltigen Gemeinwohls“? – und wenig fundierte Kritik oder konkret-konstruktive Ideen, dafür ist es jedoch unterschrieben von allen relevanten Personen aus der Kritiker*innen-Szene. Andererseits ist sie als Person nun bekannter, bekommt viel mehr Medienanfragen als vorher und kann somit ihre Arbeitsergebnisse und Ideen viel besser medial verbreiten.

Sie liest mit Interesse, dass einige ihrer aktivistischen Mitstreiter*innen in der Zeit, in der sie für das AI&I4F-Projekt gearbeitet hatte, an Gesetzgebungs-Konsultationen öffentlicher Stellen sowohl auf nationaler Ebene als auch auf EU-Ebene teilgenommen hatten, also direkt als Teil politisch-legislativer Prozesse – und sie hatte nicht einmal auf die E-Mails ihrer alten Clique reagiert. Sicherlich sind diese Eingaben weit weniger medial beachtet worden, aber vielleicht wirken sie im Hintergrund, etwa weil EU-Parlament und Bundesregierung um Einschätzungen zum Thema der Regulierung von Digitalisierung allgemein und KI im Speziellen hinsichtlich Nachhaltigkeit gebeten hatten, was ja genau ihr Bereich gewesen wäre. Sie ist unsicher, wie sie das alles beurteilen und sich zukünftig zu solchen Initiativen wie AI&I4F verhalten soll, denn auch die Konsultationen werden natürlich nicht direkt umgesetzt. In der Nacht fällt Hannah wieder ein, was sie Jelena vor zwei Jahren noch gesagt hatte: „Aktivist*innen und auch Wissenschaftler*innen sollten sich gut überlegen, wofür sie ihre kostbare Zeit verwenden – und wofür eben nicht.“

Fragen

1. Wie sind solche Veranstaltungen zu bewerten, sind sie nötig für einen sinnvollen gesellschaftlichen Diskurs? Welche Gefahren liegen darin?
2. Das ganze Schloss wirkt nachhaltig gestaltet, reproduziert aber Klischees und scheint sehr künstlich zu sein. Ist das im Sinne einer guten Arbeitsatmosphäre zu befürworten?
3. Wie hätte Hannah die Treffen in ihrem Sinne produktiver gestalten können?
4. Ist es nicht besser, mehr Medieninteresse zu wecken, als thematisch tief im eigenen Kämmerlein zu arbeiten? Ist Hannah überkritisch was das angeht?
5. Welche Formen demokratischer Partizipation sind zu befürworten, welche Kriterien könnten dafür angelegt werden und fielen die AI&I4F-Aktivitäten darunter?
6. Unabhängig vom Fallbeispiel: Bei welchen gesellschaftlichen Themen wurde jüngst nach „Ethik“ und „Werten“ gerufen? War das Themenfeld tatsächlich neu und unbekannt, sodass es gesellschaftlich diskutiert werden muss oder aber sollte durch den Aufruf eine Regulierung verhindert werden?
7. Braucht es einen Ethik-Kodex für Ethik-Kodizes, also Leitplanken und rote Linien für Ethikleitlinien, damit sie auch einen Beitrag liefern und nicht nur als Feigenblatt missbraucht werden?

Informatik Spektrum (2021) 44:134–135
https://doi.org/10.1007/s00287-021-01345-z

FORUM

E-ID-Gesetz und Datenschutz

Ursula Sury[1]

Online publiziert: 16. Februar 2021

Ausgangslage

In der Herbstsession 2019 hat das Parlament das Bundesgesetz über elektronische Identifizierungsdienste (BGEID oder E-ID-Gesetz) verabschiedet. Damit soll die Handhabung elektronischer Identitäten (E-ID) geregelt werden. Gegen das E-ID-Gesetz wurde das Referendum ergriffen und am 7. März 2021 findet die Volksabstimmung statt.

Die elektronische Identität soll das Leben der Internetnutzerinnen und -nutzer in der Schweiz einfacher machen. Mit einem einzigen Login können sie sich auf den unterschiedlichsten Websites anmelden. Vergessene Passwörter sollen der Vergangenheit angehören. Doch die E-ID kann noch mehr, sie kann auch die Identität oder das Geburtsdatum der Nutzenden nachweisen. Damit lässt sich zum Beispiel online ein Arzttermin vereinbaren, eine Behörde kontaktieren oder ein Whisky kaufen. Die staatlich anerkannte E-ID soll dabei garantieren, dass niemand Missbrauch betreibt oder sich als andere Person ausgibt. Wer im Internet Waren oder Dienstleistungen bezieht, muss sich identifizieren. Dafür gibt es verschiedene Verfahren, oft mit Benutzername und Passwort. Aber keines dieser Verfahren ist in der Schweiz gesetzlich geregelt und für keines übernimmt der Bund eine Garantie, wonach es sicher und zuverlässig funktioniere.

Wie wird die E-ID angewendet?

Die Verwendung einer E-ID ist freiwillig. Wer eine nutzen will, stellt zuerst bei einer vom Bund anerkannten E-ID-Anbieterin einen Antrag. Die Anbieterin übermittelt den Antrag an den Bund, der die Identität der antragstellenden Person prüft und der Anbieterin grünes Licht für die Ausstellung der E-ID gibt. Bund und Parlament gehen davon aus, dass es mehrere Anbieterinnen geben wird, die miteinander im Wettbewerb stehen werden. Es ist möglich, sich bei mehreren Identity-Providern gleichzeitig anzumelden.

Das E-ID-Gesetz legt fest, dass der Staat die Hoheit über den Identifizierungsprozess behält. Wie beim Pass oder der Identitätskarte, die durch akkreditierte Unternehmen hergestellt werden, ist der Staat weiterhin für die amtliche Bestätigung einer Identität zuständig. Die technische Infrastruktur hingegen wird von privaten Unternehmen entwickelt und betrieben. Diese Aufgabenteilung ist zweckmässig. Indem das bestehende Know-how von schweizerischen Unternehmen genutzt wird, lässt sich die E-ID rasch umsetzen, ohne dass die Kontrollfunktion des Staates geschwächt oder hoheitliche Rechte beschränkt werden. Der Bund reguliert und kontrolliert die Anbieter von E-ID-Lösungen. Um sich als so genannter Identity-Provider zertifizieren zu können, müssen Unternehmen eine Reihe von Kriterien erfüllen. Auch müssen die Daten in der Schweiz gespeichert und bearbeitet werden.

Nicht nur für die Bürger, auch bei den Behörden führt das neue Gesetz zu Vereinfachungen. So hängt zum Beispiel die erfolgreiche Einführung des elektronischen Patientendossiers oder eine medienbruchfreie digitale Steuererklärung massgebend von einer sicheren und damit akzeptierten elektronischen Identität ab. Die Kantone und Gemeinden sind noch stärker als der Bund auf den direkten Kontakt mit den Bürgern ausgerichtet und daher in besonderem Masse an einer E-ID-Lösung interessiert. Zudem zeigt die aktuelle Coronakrise, dass eine verlässliche E-ID einen wichtigen Beitrag zur Reduktion von Behördengängen einzig zum Zweck der Unterzeichnung leisten kann. Solche Vereinfachungen haben letztlich auch einen positiven Effekt auf die kantonalen Finanzen.

Datenschutzrechtliche Risiken

Bei der Ausstellung und der Nutzung der E-ID können, wie bei jedem Identifizierungsverfahren, sensible Daten der Userinnen und User missbräuchlich verwendet werden. Im

✉ Ursula Sury
ursula.sury@hslu.ch

[1] Luzern, Schweiz

Folgenden werden einige Risiken aufgezeigt, welche sich insbesondere bezüglich des Datenschutzes ergeben können.

Mit dem E-ID-Gesetz besteht ein Risiko, dass die Datenweitergabe seitens der E-ID-verwendenden Dienste, insbesondere innerhalb eines Konzerns, nicht genau geregelt wird. Wohl definiert das Datenschutzgesetz in Art. 4 Abs. 3 den Grundsatz der Zweckbindung und die Tatsache, dass der Verwendungszweck der Datenbeschaffung feststehen und ersichtlich sein muss, jedoch kann dies aber mittels Datenschutzerklärung bei der Erstregistrierung relativ einfach umgangen werden.

In den Privacy-Statements könnten dann Aussagen wie: „Die Einwilligung zur Datenweitergabe innerhalb des Konzerns zwecks Verbesserung der eigenen Produkte wird erteilt." oder „Der Verwendung der persönlichen Daten zur Steigerung der Nutzererfahrung innerhalb des Konzerns wird zugestimmt." Gerade Letzteres wird meistens dazu benutzt, Datenauswertungen zu erstellen, um beispielsweise die Suchergebnisse zu verfeinern.

Das E-ID-Gesetz macht des Weiteren keine Aussagen darüber, ob Daten in der Cloud gespeichert werden dürfen, wenn die Betreiberin von hinreichenden Schutzgarantien durch Verträge im Ausland ausgeht (Art 6. Abs. 2 lit a. DSG).

Datenschutzrechtliche Chancen

Die datenschutzrechtlichen Vorteile der selbstbestimmten Identität auf der Blockchain sind demgegenüber vielversprechend. Die Datensicherheit ist gewährleistet und unzulässige Zugriffe auf die Daten werden erschwert bis verunmöglicht. Die Zugriffe erfolgen transparent und können dadurch rückverfolgt werden. Die Daten fliessen zudem nur in einem begrenzten Masse, da der Nutzende die Menge kontrollieren kann. Dies kommt dem Verhältnismässigkeits-Prinzip der Datenbearbeitung zugute, wonach nur die für einen bestimmten Zweck erforderlichen Daten bearbeitet werden sollen. Die Daten sind zudem vor Verfälschungen geschützt, da für eine Änderung oder eine Löschung der Daten die Zustimmung nötig wäre. Nicht zu vergessen ist u. a. auch die Datenportabilität. Die Benutzenden erhalen dadurch die Möglichkeit, ihre personenbezogenen Daten bei einem Anbieterinnenwechsel mitzunehmen, was das Recht des Einzelnen auf informationelle Selbstbestimmung stärkt. Auch aus Sicht der Online-Anbieterinnen gibt es einen wichtigen Vorteil zu nennen. Bisher bestand eine gewisse Unsicherheit darüber, ob eingeloggte Nutzende auch tatsächlich Eigentümerinnen bzw. Eigentümer der Logindaten sind. Die SSI-Zertifikate werden beim SSI-Modell ausschliesslich der berechtigten Person zugeteilt und in deren „Wallet" gespeichert, sodass nur die jeweilige Eigentümerin/der jeweilige Eigentümer sie verwenden kann. Durch die SSI lassen sich Nutzende somit unverfälscht identifizieren.

Fazit

Mit einem Reisepass oder einer Identitätskarte kann eine Person ihre Identität im Alltag beweisen. Im Internet ist dieser Beweis derzeit nur sehr umständlich zu erbringen. Daher braucht es für die digitale Welt einen elektronischen Identitätsnachweis, auch E-ID genannt. Solche staatlich anerkannten elektronischen Identifizierungsmittel sind für die weitere Entwicklung von Online-Geschäften und E-Government-Anwendungen wichtig.

E-ID-Anbieterinnen könnten jedoch personenbezogene Daten der Nutzenden speichern und auswerten. Der Persönlichkeitsschutz der Nutzenden würde dadurch tangiert werden und ist daher zweckgebunden zu schützen.

Informatik Spektrum (2021) 44:136–137
https://doi.org/10.1007/s00287-021-01344-0

FORUM

Verifizierter Interessenskonflikt

Einsichten eines Informatikers von geringem Verstande

Reinhard Wilhelm[1]

Online publiziert: 5. Februar 2021

Eines der Lieblingslieder meiner Jugendzeit war, „If you can't be with the one you love, love the one you're with." Das klingt eher peinlich, und der geneigte Leser fragt sich, warum das hierher gehört. Aber vor einiger Zeit hörte ich eine Version dieser Aussage, die den Informatiker angeht. Ein Vizepräsident einer Firma, die sich mit der Entwicklung von fliegenden Objekten beschäftigt, sagte auf einem Workshop zum Flugwesen: „The non-existence of a verification method corresponding to a design method will not prevent us from using this design method.", um dann mit Blick auf anwesende Mitarbeiter der nationalen Flugsicherheitsbehörde einen weiteren aufschlussreichen Satz nachzuschieben: „In the end, we will get everything certified." Der erste Satz erschütterte mich etwas, den zweiten konnte ich wegen politischer Naivität nicht verstehen. Neuerdings haben uns Leaks über die Haltung der Ingenieure in dieser Firma aufgeklärt, „We sell planes that are designed by clowns and controlled by monkeys." Das trifft unsere Erkenntnisse schon ziemlich gut, außer dass wir erfahren mussten, dass die Affen, welche die Flugzeuge entwerfen, und die Clowns, die sie kontrollieren, bei der gleichen Firma arbeiten.

Die beiden Sätze werfen viele Fragen bezüglich der Sicherheit von Flugzeugen, insbesondere der Qualität des Zertifizierungsprozesses auf. Das obige Statement des Vizepräsidenten steht in einem gewissen Widerspruch zur Aussage eines Sprechers des gleichen Unternehmens, „Safety is a core value for everyone at ....". Nun könnte man sagen, Sicherheit ist Sicherheit, egal ob verifiziert oder nicht verifiziert, wenn nur nicht ein paar Abstürze Zweifel an der Sicherheit der Flieger geweckt hätten.

Eine Kommission des Joint Authorities Technical Review (JATR) hat der amerikanischen Flugsicherheitsbehörde, Federal Aviation Administration (F.A.A.), den Bericht einer international besetzten Untersuchungskommission über die Gründe für die Abstürze mehrerer Flugzeuge vom Typ Boeing 737 Max und die Defizite bei der Zertifizierung dieses Flugzeugtyps übergeben. Eine Kommission des amerikanischen Repräsentantenhauses kam in ihrem Bericht vom September 2020 zu ähnlichen Erkenntnissen. Zu Ihrer Erinnerung: Die Boeing 737 ist ein schon lange und in über 10.000 Exemplaren um die Welt fliegendes Flugzeug. Es wurde bereits 1967 zertifiziert. Um den Treibstoffverbrauch zu reduzieren, baute Boeing ein neues Triebwerk an den bewährten 737-Rumpf. Leider veränderte das die Flugeigenschaften stark. Das neue Design wurde Boeing-intern, wie schon oben gesagt, als von Clowns entworfen bezeichnet. Um die veränderten Flugeigenschaften zu kompensieren, mussten Informatiker ran. Mit Software lässt sich ja bekanntlich alles reparieren. Die Software, das Maneuvering Characteristics Augmentation System (MCAS), steuerte den Neigungswinkel des Flugzeugs – wie wir durch zwei Abstürze wissen, auf sehr fehleranfällige Weise. Der Euphemismus, *Steigerung der Manöverierungscharakteristik*, ist nicht ganz untypisch für Boeing. Es soll bei Boeing einen für die Erfindung von Euphemismen zuständigen hochrangingen Mitarbeiter gegeben haben. Auch der Name *Dreamliner* geht auf sein Konto. Wenn man sich neuere Veröffentlichungen zu seinen Mängeln anschaut, könnte er zum *Nightmareliner* werden. So haben Boeing-Entwickler der Öffentlichkeit gesteckt, dass bei Druckverlust in der Kabine in nur drei von vier Tests die Sauerstoffmasken herunterfielen. Da sollte man den entsprechenden Text bei den Routinesicherheitsanweisungen vielleicht so ändern: „In the unlikely event of a loss of cabin pressure, panels above your seat might open and oxygen masks might fall from the overhead compartment. Make sure that you don't get hit!"

Der Bericht der oben genannten Kommission betrachtete nur die Zertifizierung des Flight Control Systems der Boe-

✉ Reinhard Wilhelm
wilhelm@cs.uni-saarland.de

[1] Saarland Informatics Campus, Saarbrücken, Deutschland

ing 737 Max. Sie wurde 2017 in einem vereinfachten und weniger strengen Verfahren zertifiziert. Boeing argumentierte, und die F.A.A. akzeptierte, dass der neu entworfene Flieger auf einer langen bewährten Baureihe von 737-Flugzeugen beruhte. Die durch den Einbau neuer Triebwerke massiv gegenüber früheren 737-Fliegern veränderten Flugeigenschaften störten diese Argumentation nicht. Schließlich wurden Boeing-intern die Kontrolleure der F.A.A. als Affen bezeichnet, und was stört schon einen Affen?

Die wesentliche Erkenntnis der Kommission war, dass die F.A.A. gar nicht wusste, was Boeing da entwickelte und anschließend zertifiziert bekommen wollte. Die F.A.A. wusste zwar, dass Boeing an MCAS arbeitete, und was MCAS tun sollte. Sie bekam jedoch von Boeing die Information nur stückweise und an verschiedene Adressaten innerhalb der F.A.A. geliefert. Außerdem wurde sie von Boeing nicht informiert, dass das Design massiv während des Entwicklungsprozesses geändert wurde, und MCAS dadurch viel stärker in das Flugverhalten eingreifen konnte, als ursprünglich geplant. Wenn man es genau bedenkt, keine gesunde Situation für eine sachgemäße Zertifizierung.

Außerdem mutmaßte die Kommission, dass es eventuell bei den Boeing-Mitarbeitern, welche im Auftrag der F.A.A. an der Zertifizierung dieses Boeing-Flugzeugs arbeiteten, Interessenskonflikte gegeben haben könnte. Ja, das ist mal eine Überraschung! Wer wäre darauf gekommen?

Klar wurde, dass MCAS in der Standardversion der Boeing 737 Max Daten nur von einem Sensor bekam. Wenn der ausfiel, sah es schlecht aus. In einer Premium-Version kriegt der Kunde zwei Sensoren, die ihre Werte abgleichen. Gerüchteweise verlautete, dass Boeing auch für die einfache Version Redundanz postulierte, weil nämlich bei aufeinanderfolgenden Flügen immer der Sensor gewechselt wurde, den MACS abfragte. Na, das nenne ich mal eine saubere Redundanz!

Eine Untersuchung durch die New York Times hat gezeigt, wie geschickt die amerikanische Industrie, darunter führend Boeing, die Gesetzgebung über die Zertifizierung von sicherheitskritischen Systemen beeinflusst hat. Mit dem Argument, die Bürokratie zu verringern und vor allem, um einen Wettbewerbsvorteil gegenüber der europäischen Konkurrenz zu erreichen, dürfen gemäß dieser Gesetzgebung die Hersteller die Prüfung ihrer eigenen Produkte teilweise gleich selbst vornehmen. Leuchtet ein! Reduziert die Bürokratie tatsächlich! Das war bei einer Regierung, in der Flachköpfe den tiefen Staat zurückbauen wollten, höchst willkommen.

Also fassen wir zusammen: Boeing hat durch geschickte Lobbyarbeit dafür gesorgt, dass es seine eigenen Produkte selbst zertifizieren kann, die damit beauftragten Boeing-Mitarbeiter fanden die Boeing-Produkte – ohne jeden Interessenskonflikt – einfach schnuckelig, und die F.A.A. kannte zwar den Namen MCAS und ahnte so grob, worum es ging, wusste aber nicht Genaueres, und wenn, dann so verteilt, dass kein F.A.A.-Mitarbeiter ein Gesamtbild hatte.

Ich möchte noch hinzufügen, dass Zertifizierung durchaus auch ihre negativen Seiten hat. Zum Beispiel führt sie eine gewisse Trägheit in die Entwicklung ein; ist ein System erst einmal mit großem Aufwand zertifiziert, dann scheut man den Aufwand für die Rezertifizierung, auch wenn das System nachträglich als fehlerhaft oder verbesserungswürdig erkannt wird. Die inhärente Logik möchte ich in einer Analogie zur inhärenten Logik der amerikanischen Außenpolitik beschreiben. Die pflegt auf den Vorwurf, „This ruler is a crook.“ zu antworten, „But he is our crook!“. Ähnlich ist die Antwort auf die Feststellung, „This system is shit.“ typischerweise die Antwort, „But it is certified shit!“

Informatik Spektrum (2021) 44:138–139
https://doi.org/10.1007/s00287-021-01334-2

FORUM

Um etliche Ecken ged8

Gehirn-Jogging auf Basis der math.- und informatisch-orientierten Rechtschreibreform

Rolf Windenberg[1]

Online publiziert: 25. Januar 2021

Regeln:

- Verwendung einfacher math. Symbole und von Zahlen
- Aussprache von Großbuchstaben wie im Alphabet
- Bei der informatisch-orientierten Reform: zusätzlich Verwendung programmiersprachlicher Symbole

Beispiel: „O ∙8" steht für „Oma lacht"

Knobelaufgaben:

- *Anfänger:* **gr+1tLung**
- *Spielerei mit Buchstaben:*
  Dr hR BhüT dich auf ∀n d1N WgN
- *Fortgeschrittene:* **i(S Gschah)n dR n8**
- *Experten:* **1 klapp1.orch biss si1 b1**
- *Genies:* **das 1.e ist 1e -sinn** $\binom{e}{legung}$;
  das an3 $\binom{st\ \forall s}{aus}$ **gut/(d8)**
- *Informatisch-orientiert:*
  call **wahl**

Abb. 1 Regeln und Knobelaufgaben mit kleiner Hilfe zur Lösung der vierten Knobelaufgabe [1]. Die Auflösungen finden sich im nächsten Heft

✉ Rolf Windenberg
wolfinger@informatik.uni-hamburg.de

[1] Fachbereich Informatik, Universität Hamburg, Hamburg, Deutschland

Auflösungen aus dem letzten Heft (1/2021):

- Wachtel [wegen: w-*achtel*]
- dies folgt aus der einhelligen Meinung aller Experten [wegen: d-*I*-s-*folgt aus*-*D*-r-*ein*-h-*L*-ig-*N*-m-*ein*-ung-*alle*-r-exp-*R*-t-*N*]
- videoüberwacht [wegen: vi-*D*-o-*über*-w-*acht*]
- das Echslein stolperte über ein Steinchen [wegen: da-*sechs*-l-*eins*-tolp-*R*-*T*-*über*-*eins*-t-*ein*-ch-*N*]
- Ben erstach Tiere ohne Grund [wegen: b-*N*-*erst*-*acht*-ie-re-*ohne*-gr-*und*]
- unbedingte Mieterhöhung [wegen: un-*bedingte*-m-*I*-*T*-*Erhöhung*]

Literatur

1. Windenberg R, Hasselfang RW (2018) Um etliche Ecken ged8 (Version 2.0). Shaker Media, Düren

Informatik Spektrum (2021) 44:140–145
https://doi.org/10.1007/s00287-021-01348-w

MITTEILUNGEN

Mitteilungen der GI im Informatik Spektrum 2/2021

Online publiziert: 3. März 2021

Aus Vorstand und Präsidium

Beschlüsse aus der Präsidiumssitzung vom 28./29. Januar 2021

Folgende Personen werden bestätigt, bzw. benannt:

- Francis Baud als Vertreter der Schweizer Informatik Gesellschaft
- Prof. Dr. Heribert Vollmer (Leibniz Universität Hannover) als Sprecher des FB GInf
- Prof. Dr. Christian Scheideler (Universität Paderborn) als stellvertretender Sprecher des FB GInf
- Prof. Dr. Jörg Desel als Sprecher des FB IAD
- Prof. Dr. Jan Vahrenholt als stv. Sprecher des FB IAD
- Prof. Dr.-Ing. Hans-Joachim Hof, TH Ingolstadt, RG Ingolstadt als Regionalgruppenvertretung im Präsidium
- Manuel Friedrich, DLR Projektträger Berlin, RG Berlin-BB als Regionalgruppenvertretung im Präsidium
- Prof. Dr. Andreas Oberweis als Hauptherausgeber der „Lecture Notes in Informatics“ (LNI)
- Prof. Dr. Agnes Koschmider als Herausgeberin der Reihe „Thematics“ in den LNI
- Dr. Judith Michael als Herausgeberin der Reihe „Seminars“ in den LNI

Folgende Präsidiumsarbeitskreise werden für ein weiteres Jahr bestätigt:

- PAK Datenschutz und IT-Sicherheit
- PAK eScience
- PAK Grand Challenges

Folgende Präsidiumsverantwortliche werden für ein weiteres Jahr bestätigt:

- CEPIS: Prof. Dr. Kai Rannenberg
- Hauptherausgeber für das „Informatik Spektrum“: Prof. Dr. Thomas Ludwig
- Past President: Prof. Dr. Peter Liggesmeyer
- GI-Dissertationspreis: Prof. Dr. Steffen Hölldobler
- Max-Planck-Gesellschaft (MPG): Prof. Dr. Andreas Oberweis
- Vertreter im Stiftungsrat der Stiftung „Werner-von-Siemens-Ring“: Prof. Oliver Günther, Ph.D.
- Anerkennungsausschuss Medizinischer Informatikerinnen und Informatiker von GI, GMDS und BVMI: Prof. Dr. rer. nat. Dipl.-Ing. Thomas Deserno
- DIN: Prof. Dr. Kai Rannenberg
- CHE: Prof. Dr. Gregor Engels
- acatech: Prof. Dr. Peter Liggesmeyer

Folgende Personen werden in den Haushaltsausschuss gewählt:

- Prof. Dr. Kai Rannenberg
- David Richter
- Prof. Dr. Maria Wimmer

Anträge von Mitgliedern:

- Die vorgelegten Anträge werden nicht vom Präsidium in die Ordentliche Mitgliederversammlung eingebracht.

Aus der Geschäftsstelle

Wir danken unseren Fördermitgliedern.

- Accso – Accelerated Solutions GmbH, Darmstadt
- acocon GmbH, Bielefeld
- ADB Solutions GmbH, Bindlach
- Advanced Dynamics GmbH, Köln
- AKAD Bildungsgesellschaft mbH, Stuttgart
- AKDB – Anstalt für Kommunale Datenverarbeitung in Bayern, München
- Alfred-Wegener-Institut, Bremerhaven
- A•S Technology Consulting GmbH, Frankfurt am Main
- Aschert & Bohrmann GmbH, Köln
- ask-Innovative Visualisierungslösungen GmbH, Darmstadt
- Asteas Technologies GmbH, Landeck

- Bayer Business Services GmbH, Leverkusen
- BDE Business Datawarehouse Engineering GmbH, Berlin
- BGfD Bayreuther Gesellschaft für Datenschutz mbH, Bayreuth
- BITBW, Stuttgart
- BESIT e. V., Nürnberg
- Beuth Hochschule für Technik Berlin, Berlin
- Bildungszentrum für informationsverarbeitende Berufe e. V., Paderborn
- bluecue consulting GmbH & Co.KG, Bielefeld
- Boldly Go Industries GmbH, Frankfurt am Main
- boTec GmbH, Wiesbaden
- BRAINGINES SA, Fribourg
- BSSM Becker Software System Management GmbH, Rossdorf
- Büren & Partner, Nürnberg
- Bundesamt für Sicherheit in der Informationstechnik, Bonn
- Bundesverband IT-Mittelstand e. V. (BITMi), Aachen
- Capgemini Deutschland GmbH, München
- Carl-Cranz-Gesellschaft e. V., Wessling/Obb.
- CAST e. V., Darmstadt
- CertEuropA GmbH, Kassel
- CGI Deutschland BV. & Co. KG, Braunschweig
- CODE University of Applied Sciences, Berlin
- CommitWork GmbH, Dortmund
- con terra GmbH, Münster
- creatale GmbH, Ludwigsburg
- Dataport, Altenholz
- Der Hessische Beauftragte für Datenschutz und Informationsfreiheit, Wiesbaden
- Deutsche Hochschule für Prävention und Gesundheitsmanagement, Saarbrücken
- Deutsches Forschungszentrum für künstliche Intelligenz GmbH, Kaiserslautern
- DHBW Heidenheim, Heidenheim
- Duale Hochschule Baden-Württemberg, Stuttgart
- Duale Hochschule Baden-Württemberg, Stuttgart
- Duale Hochschule Baden-Württemberg, Karlsruhe
- Duale Hochschule Gera-Eisenach, Gera
- edv-anwendungsberatung zühlke & bieker gmbH, Recklinghausen
- eifel-online GmbH, Mechernich
- Fachbereich DCSM, Wiesbaden
- Fachhochschule Aachen, Aachen
- Fachhochschule für die Wirtschaft Hannover, Hannover
- Fachhochschule Dortmund, Dortmund
- Fachhochschule Münster, Münster
- Fachhochschule Schmalkalden, Schmalkalden
- Fachhochschule Wedel, Wedel
- FINCON Unternehmensberatung GmbH, Hamburg
- FIZ Karlsruhe GmbH, Eggenstein-Leopoldshafen
- Forschungszentrum Informatik, Karlsruhe
- Forschungszentrum Jülich GmbH, Jülich
- FOKUS Fraunhofer Institut für Offene Kommunikationssysteme, Berlin
- Fraunhofer-Gesellschaft e. V., München
- Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e. V., Wachtberg
- Fraunhofer IESE, Kaiserslautern
- Fraunhofer-Institut für Kognitive Systeme IKS, München
- Fraunhofer Institut IAO, Stuttgart
- Freies katholisches Gymnasium St. Meinrad, Rottenburg a.N.
- Friedrich-Schiller-Universität, Jena
- Geib IT GmbH, Saarwellingen
- G.E.M. Team Solutions GmbH & Co. KG, Neustadt
- Generali Deutschland Informatik Services GmbH, Aachen
- genua gmbh, Kirchheim
- Georg-August-Universität Göttingen, Göttingen
- Gesamtschule Bad Lippspringe Schlangen, Bad Lippspringe
- Gesellschaft der Förderer u. Freunde der HS Würzburg-Schweinfurt e. V., Würzburg
- Gesellschaft für Datenschutz und Datensicherheit e. V., Bonn
- GESIS Gesellschaft für Informationssysteme mbH, Salzgitter
- GESIS – Leibniz-Institut für Sozialwissenschaften, Köln
- GFU Cyrus AG, Köln
- Goerigk Informationstechnologie GmbH, Bad Nauheim
- Graf-Stauffenberg-Gymnasium, Flörsheim a.M.
- GWDG – Gesellschaft für wissenschaftliche Datenverarbeitung mbH, Göttingen
- Gymnasium Wolbeck, Münster
- Handelsblatt GmbH, Düsseldorf
- Hasso-Plattner-Institut für SW-Systemtechnik GmbH, Potsdam
- HAW Hochschule Amberg-Weiden, Amberg
- HDI Systeme AG, Hannover
- Heinrich-Heine-Gymnasium, Dortmund
- Heinrich-Heine-Gymnasium, Hamburg
- Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Berlin
- Helmholtz-Zentrum Geesthacht, Geesthacht
- Helmholtz-Zentrum Potsdam, Potsdam
- Helmut Schmidt Universität, Hamburg
- Hessische Zentrale für Datenverarbeitung, Wiesbaden
- HIS Hochschul-Informations-System eG, Hannover
- HMS Analytical Software GmbH, Heidelberg
- HNF Heinz Nixdorf MuseumsForum GmbH, Paderborn
- Hochschule Aalen, Aalen
- Hochschule Albstadt-Sigmaringen, Albstadt

- Hochschule Anhalt, Köthen
- Hochschule Bremerhaven, Bremerhaven
- Hochschule der Bildenden Künste Saar, Saarbrücken
- Hochschule des Bundes für Öffentliche Verwaltung, Brühl
- Hochschule Esslingen, Esslingen
- Hochschule Fulda, Fulda
- Hochschule Furtwangen, Villingen-Schwenningen
- Hochschule Hannover, Hannover
- Hochschule Karlsruhe, Karlsruhe
- Hochschule Niederrhein, Krefeld
- Hochschule Ruhr West, Bottrop
- Hochschule Weserbergland, Hameln
- Hochschule Wismar, Wismar
- Hönne-Berufskolleg, Menden
- HPA Hamburg Port Authority AöR, Hamburg
- Humboldt Universität Berlin, Berlin
- IBM Deutschland GmbH, Ehningen
- Informatik Consulting Systems GmbH (ICS GmbH), Stuttgart
- imbus AG, Möhrendorf
- Immanuel-Kant-Gymnasium Bad Oeynhausen, Bad Oeynhausen
- Infodas Gesellschaft für Systementwicklung und Informationsverarbeitung mbH, Köln
- Informatik Akademie, BP V 205 Abidjan
- Information und Technik NRW, Düsseldorf
- Information Works GmbH, Köln
- Initiative D21 e. V., Berlin
- innoQ Deutschland GmbH, Monheim am Rhein
- Institut für Ökonometrie, Bonn
- Institutsverbund Informatik, Stuttgart
- IT manufaktur GmbH, Bad Schwalbach
- IT-Forum Rhein-Neckar e. V., Ludwigshafen
- IT Service Omikron GmbH (ITSO), Berlin
- ITech Progress GmbH, Ludwigshafen
- iteratec GmbH, München
- IVU Traffic Technologies AG, Berlin
- jambit GmbH, München
- Karlsruher Institut für Technologie – KIT, Eggenstein-Leopoldshafen
- Landeshauptstadt München, München
- Lessing-Gymnasium Karlsruhe, Karlsruhe
- Login-IT GmbH & Co. KG, Kirchheimbolanden
- Max-Planck-Institut für Informatik, Saarbrücken
- mehrwert intermediale kommunikation GmbH, Köln
- Menzel IT GmbH, Berlin
- MicroConsult Microelectronics Consulting & Training GmbH, München
- msg systems ag, Ismaning
- MSVH GmbH & Co. KG, Bremerhaven
- NASS GmbH, Hannover
- Netzkunst24 Design- und Internetagentur GmbH, Lüneburg
- Neubert Consulting GmbH, Fürth
- NORDAKADEMIE, Elmshorn
- Normfall GmbH, München
- NovaTec Consulting GmbH, Leinfelden-Echterdingen
- OFFIS e. V., Oldenburg
- Opitz Müller und Partner GbR, Berlin
- OSSENO Software GmbH, Kaiserslautern
- Ostfalia Hochschule für angewandte Wissenschaften, Wolfenbüttel
- Otto-Friedrich-Universität Bamberg, Bamberg
- Otto-von-Guericke-Universität Magdeburg, Magdeburg
- Patzschke + Rasp Software GmbH, Wiesbaden
- perbit Software GmbH, Altenberge
- PPI AG, Hamburg
- Private FernFachhochschule Darmstadt, Darmstadt
- PROMATIS software GmbH, Ettlingen
- QAware GmbH, München
- Qualmity, Berlin
- Rohde & Schwarz GmbH & Co. KG, München
- RWTH Aachen (IMA), Aachen
- SD Software-Design GmbH, Bad Krozingen
- SAP SE, Berlin
- Schulverein Anna-Schmidt e. V., Frankfurt
- Seeliger & Co. GmbH, Eichenau
- SHD Einzelhandelssoftware GmbH, Andernach
- SieITMCi Siebenhofer.Consulting e.U., Steyr
- Siemens AG, München
- SIZ GmbH, Bonn
- SMO GmbH, Karlsruhe
- snoopmedia GmbH, Bonn
- SoftServe UGE GmbH, Frankfurt
- Software AG, Darmstadt
- Softwarekontor GmbH, Ludwigshafen
- Splendid Minds GmbH, Fürth
- Springer-Verlag GmbH, Heidelberg
- SQLan Gesellschaft für Informations- und Netzwerksysteme mbH, Wiesbaden-Sonnenberg
- SRH Hochschule Heidelberg, Heidelberg
- Synedat Consulting GmbH, Helmstedt
- SYRACOM AG, Wiesbaden
- SYSLAB.COM GmbH, München
- Talent.io Recruitment GmbH, Berlin
- TBiQ Managed Testing Services GmbH, Bochholt
- Technische Akademie Esslingen e. V., Ostfildern
- Technische Universität Braunschweig, Braunschweig
- Technische Universität Chemnitz, Chemnitz
- Technische Universität Dresden, Dresden
- Technische Universität München, Garching
- TH Brandenburg University of Applied Sciences, Brandenburg
- TH Mittelhessen, Gießen

- The MathWorks GmbH, Ismaning
- Transaction Software GmbH, München
- TU Berlin, Berlin
- TU Dortmund, Dortmund
- TÜV Informationstechnik GmbH, Essen
- TrendTec UG (haftungsbeschränkt), Edingen-Neckarhausen
- Union IT-Services GmbH, Frankfurt
- Universität Bamberg, Bamberg
- Universität Bremen, Bremen
- Universität der Bundeswehr München, Neubiberg
- Universität des Saarlandes, Saarbrücken
- Universität Duisburg-Essen, Essen
- Universität Erlangen-Nürnberg, Erlangen
- Uni HH/MIN Fakultät/FB Informatik, Hamburg
- Universität Hamburg, Hamburg
- Universität Kassel, Kassel
- Universität Koblenz-Landau, Koblenz
- Universität Leipzig, Leipzig
- Universität Mannheim, Mannheim
- Universität Osnabrück, Osnabrück
- Universität Paderborn, Paderborn
- Universität Rostock, Rostock
- Universität Tübingen, Tübingen
- University of Digital Science Berlin, Berlin
- UXMA GmbH & Co. KG, Kiel
- Verband der Vereine Creditreform e. V., Neuss
- VOSDAV GmbH, Berlin
- VRnow GmbH, Berlin
- Walter de Gruyter GmbH, München
- Werum IT Solutions GmbH, Lüneburg
- Westfälische Hochschule Gelsenkirchen, Gelsenkirchen
- zisa consulting GmbH, Berlin
- ZKM Zentrum für Kunst und Medien, Karlsruhe

Presse- und Öffentlichkeitsarbeit der GI

Datenstrategie der Bundesregierung: Datenkompetenz braucht informatische Bildung für alle (28.01.2021)

Kurz vor Ende der Legislaturperiode hat die Bundesregierung ihre lange angekündigte Datenstrategie veröffentlicht. Während an vielen Stellen konkrete Maßnahmen verabredet werden, beinhaltet die Strategie zur grundlegenden Frage der Vermittlung von Datenkompetenzen nur wenige Maßnahmen und vernachlässigt die Informatik als die Wissenschaft für die automatisierte Verarbeitung von Daten.

Prof. Dr. Ira Diethelm, Mitglied im GI-Präsidium: „Die Informatik ist nicht nur die wichtigste Bezugswissenschaft der Digitalisierung, sondern auch die Disziplin, die den Umgang mit großen Datenmengen erst möglich macht. Deshalb ist es auch beim Erwerb von Datenkompetenzen von entscheidender Bedeutung, die Prinzipien und Grenzen der automatisierten Datenverarbeitung zu verstehen. Wer datenkompetente Bürgerinnen und Bürger will, muss eine informatische Bildung insbesondere in der Schule ermöglichen. Das Schulfach Informatik leistet zur Datenkompetenz einen essenziellen Beitrag. Datenkompetenzen ohne Informatik vermitteln zu wollen ist so, als wollte man Umweltbildung ohne Naturwissenschaften betreiben."

Die Datenstrategie strebt an, dass „Alle Schülerinnen und Schüler [...] lernen, wie Daten erhoben, verarbeitet, kritisch ausgewertet und genutzt werden". Leider wird die Informatik als zentrales Fach weder namentlich erwähnt, noch wird das Ziel durch konkrete Maßnahmen unterfüttert. Damit verpasst die Bundesregierung die Chance zu einem klaren Bekenntnis für zeitgemäße Bildung in einer digital vernetzten Welt sowie für Informatik als obligatorisches Fach. Damit bleibt die Datenstrategie weit hinter den Empfehlungen vieler Fachleute u.a. des eigenen Wissenschaftsrates zurück. Im Oktober 2020 hat das wichtigste wissenschaftspolitische Beratungsgremium von Bund und Ländern empfohlen, „die schnelle und flächendeckende Einführung informatischer Bildung in den Schulen noch stärker zu priorisieren", denn sie sei ein zentraler Schlüssel, „um den digitalen Wandel in der Gesellschaft erfolgreich, inklusiv und nachhaltig zu gestalten."

Weiterhin sieht die Datenstrategie vor, dass „allen, die eine Ausbildung oder ein Studium in Deutschland abgeschlossen haben, ein Mindeststandard an Datenkompetenz vermittelt wurde" (S. 44). Auch hier fehlen konkrete Maßnahmen zur Umsetzung. Zwar fallen beide Themen in den Kompetenzbereich der Länder – den Empfehlungen des Wissenschaftsrats folgend sollte die Strategie dennoch ein klares Bekenntnis zu einem verpflichtenden Informatik-Unterricht enthalten und einen entsprechenden Koordinierungsprozess mit den Ländern vorsehen. Dies sollte in der angekündigten „Roadmap Datenkompetenzen" nachgeholt werden.

GI fordert, die digitale Souveränität von Schülerinnen und Schülern, Schulen und des IT-Standortes Deutschland zu stärken (18.01.2021)

Die Corona-Krise hat den Digitalisierungsrückstand des deutschen Bildungssystems schmerzlich vor Augen geführt. Viele Schülerinnen und Schüler, Eltern und Lehrkräfte leiden unter unzureichenden digitalen Infrastrukturen, fehlenden Softwarelösungen und mangelnden Strategien und Lehrkonzepten für den pandemiebedingten Distanzunterreicht. Vor diesem Hintergrund ist es verständlich, dass kurzfristig auf kommerzielle, datenschutzrechtlich nicht geprüfte Softwareanwendungen zurückgegriffen wurde und zum Teil auch noch immer zurückgegriffen wird.

Im Hinblick auf den akuten Handlungsbedarf war das nachvollziehbar.

Insbesondere die bundeslandweite Einführung von Software, die Daten außerhalb des Geltungsbereichs der Europäischen Datenschutzgrundverordnung (DSGVO) speichert, sieht die Gesellschaft für Informatik e. V. jedoch als bedenklich an. In Baden-Württemberg hat sich beispielsweise ein breites Bündnis gegen die geplante Einführung von MS 365 als Software-Lösung für alle Schulen gestellt.

Bereits heute existieren leistungsfähige und leicht bedienbare Cloud-Lösungen und Lernmanagementsysteme, die einen datenschutzkonformen und damit rechtssicheren Betrieb auf deutschen bzw. europäischen Servern erlauben. Beispielhaft seien hier die Lernmanagementsysteme Moodle und Ilias genannt, die mit Plugins wie H5P interaktive und kooperative Arbeitsformen unterstützen und eine geeignete Grundlage der IT-Infrastruktur von Schulen darstellen. Zuletzt konnte der reibungslose Betrieb allerdings aufgrund fehlender personeller und technischer Ressourcen nicht in allen Bundesländern gewährleistet werden.

Den kompletten Text der Pressemitteilungen finden Sie unter https://gi.de/aktuelles/presse.

Tagungsankündigungen

Jahrestagung „KI und Ethik" der Fachgruppe Frauen und Informatik am 24.04.2021

KI entwickelt sich zu einem der spannendsten und herausforderndsten Themen unserer Zeit. Sie greift in alle Lebensbereiche ein. Umso wichtiger ist es, dass wir die Entwicklungen auch unter gesellschaftlichen Aspekten und besonders unter dem Gendergesichtspunkt betrachten und gestalten. Stichworte sind Transparenz, Erklärbarkeit, Gerechtigkeit und Fairness, Freiheit und Autonomie, Vertrauen und Nachhaltigkeit.

Außerdem vergeben wir im Rahmen unserer Tagung zum zweiten Mal den Preis der Fachgruppe für herausragende Abschlussarbeiten an Informatik-Studentinnen.

Das Format der Tagung wird überwiegend virtuell sein. Aktuelle Informationen unter:

www.frauen-informatik.de

INFORMATIK 2021 – 51. Jahrestagung der Gesellschaft für Informatik in Berlin

Die INFORMATIK 2021 ist die Jahrestagung und der wichtigste Treffpunkt der Gesellschaft für Informatik e. V. (GI). Die 51. Jahrestagung INFORMATIK 2021 findet vom 27.09. bis 01.10.2021 im virtuellen Raum statt.

2021 steht die Informatik ganz im Zeichen der Nachhaltigkeit, der nachhaltigen Gestaltung der Informatik und der Nachhaltigkeit durch die Informatik. Leitbild sind die 17 Ziele für eine nachhaltige Entwicklung der Vereinten Nationen. Leitfragen der Konferenz sind: Welchen Beitrag kann die Informatik für eine nachhaltige Entwicklung leisten? Wie muss sich die Disziplin selbst aufstellen, um nachhaltiger zu agieren? Gemeinsam mit dem Fachausschuss Umweltinformatik, dem Fachbereich Künstliche Intelligenz und weiteren Gliederungen sollen Nachhaltigkeitsfragen in der Breite der Informatik diskutiert werden, dies in abwechslungsreichen Veranstaltungsformaten, von Vorträgen, Panelgesprächen und Fishbowl-Diskussionen.

Auf der INFORMATIK 2021 bringen wir Gesprächspartnerinnen und -partner aus Wissenschaft, Politik und Wirtschaft zusammen, beleuchten gesellschaftliche und politische Zusammenhänge und suchen den Dialog mit allen Interessierten. Dabei behandelt die INFORMATIK 2021 die folgenden vier Dimensionen der Informatik: ökologisch, ökonomisch, sozial und technologisch. Neben dem Schwerpunkthema Nachhaltigkeit wird es auch wieder Workshops zu aktuellen Themen der Informatik geben.

Die Jahrestagung wird von folgenden Teiltagungen begleitet:

- EnviroInfo 2021- der Fachtagung des GI- Fachausschuss Umweltinformatik
- KI 2021 – der Jahrestagung des GI-Fachbereichs Künstliche Intelligenz
- SKILL 2021 – der Studierendenkonferenz Informatik

Weitere Informationen unter https://informatik2021.gi.de/.

Personalia

Peter Federer gestorben

Der ehemalige GI-Geschäftsführer, Dr. Peter Federer, ist im November 2020 verstorben. Von 2005 bis 2015 hat er die GI geleitet. Unter seine Ägide fand das Informatikjahr 2006 unter dem Motto „dank Informatik" statt. Unser Mitgefühl gilt seinen Söhnen Sebastian und Christoph Federer und ihren Familien.

Aus den Gliederungen

Digital Health definiert

Die Fachgruppe Digital Health hat in der Enzyklopädie Wirtschaftsinformatik einen Beitrag zur Begriffsbestimmung von Digital Health veröffentlicht. In dem Eintrag betonen die Autoren zum einen die Nähe zum Verständnis von E-Health, zum anderen wird jedoch auch der Zeit-

geist hinter dem Begriff Digital Health herausgearbeitet. Demnach „[..]betont der Begriff die tiefe transformatorische Dynamik, die sich aus der ubiquitären Verfügbarkeit von Informationen und Analysekapazitäten ergibt, mit denen bspw. Vital- bzw. biometrische Werte kontinuierlich erhoben, verarbeitet sowie Trends und Risiken analysiert werden. Eng verbunden mit der allgemeinen Bewegung der digitalen Transformation, bricht das Konzept der digitalen Gesundheit radikal das Stereotyp von getrennten Gesundheitsorganisationen, die unabhängig voneinander medizinische Versorgung für Patienten anbieten.“ Unser Dank gilt den Autoren und dem Netzwerk der Fachgruppe, die sich für die Bereicherung der Enzyklopädie stark gemacht haben.

Der Beitrag ist unter folgender Quelle veröffentlicht: Schlieter H, Sunyaev A, Breitschwerdt R, Sedlmayr M (2021) Digital Health. In: Gronau N (Hrsg.) Enzyklopädie der Wirtschaftsinformatik: Online-Lexikon. GITO-Verlag, Potsdam https://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/informationssysteme/Sektorspezifische-Anwendungssysteme/Gesundheitswesen–Anwendungssysteme-im/digital-health/digital-health

GI-Tagungen

26.05.2021–28.05.2021

Bamberg
International Conference on Innovations for Community Services
I4 CS
http://i4cs-conference.org/

21.06.2021–26.06.2021

München
CARS 2021 Computer Assisted Radiology and Surgery – 35th International Congress and Exhibition
CARS 2021
https://www.cars-int.org

07.07.2021–09.07.2021

Lübeck
Medical Imaging with Deep Learning
https://2021.midl.io/

05.09.2021–08.09.2021

Ingolstadt
Mensch und Computer: Aufbruch in eine neue Zukunft
MuC 2021
https://muc2021.mensch-und-computer.de/

08.09.2021–10.09.2021

Wuppertal
GI-Fachtagung Informatik und Schule – Lehrerbildung
INFOS 2021
http://www.infos2021.de/

13.09.2021–15.09.2021

Dortmund
19. Fachtagung Bildungstechnologien der GI (gemeinsam mit der HDI 2021)
DELFI 2021
https://delfi-tagung.de/2021/delfi-2021-tagung

15.09.2021–17.09.2021

Erlangen
19. ASIM Fachtagung Simulation in Produktion und Logistik
ASIM-Fachtagung SPL
http://www.asim-fachtagung-spl.de

20.09.2021–24.09.2021

Dresden
19. Fachtagung für Datenbanksysteme für Business, Technologie und Web
BTW 2021
http://btw-konferenz.de/2021

29.09.2021–30.09.2021

Berlin
Informatik und Nachhaltigkeit
INFORMATIK2021
https://informatik2021.de