Darnstaedt, Daniel A.; Ahrens, Antje; Richter-Trummer, Valentin; Todtermuschke, Marcel; Bocklisch, Franziska

Vorgehen zur Beschreibung von menschlichem Expertenwissen und kognitiven Prozessen beim Teach-in von Industrierobotern

Zeitschrift für Arbeitswissenschaft

Provided in Cooperation with:
Springer Nature

Suggested Citation: Darnstaedt, Daniel A.; Ahrens, Antje; Richter-Trummer, Valentin; Todtermuschke, Marcel; Bocklisch, Franziska (2021) : Vorgehen zur Beschreibung von menschlichem Expertenwissen und kognitiven Prozessen beim Teach-in von Industrierobotern, Zeitschrift für Arbeitswissenschaft, ISSN 2366-4681, Springer, Berlin, Heidelberg, Vol. 76, Iss. 1, pp. 34-49, https://doi.org/10.1007/s41449-021-00284-5

Z. Arb. Wiss. (2022) 76:34–49
https://doi.org/10.1007/s41449-021-00284-5

WISSENSCHAFTLICHE BEITRÄGE

Vorgehen zur Beschreibung von menschlichem Expertenwissen und kognitiven Prozessen beim Teach-in von Industrierobotern

Daniel A. Darnstaedt[1] · Antje Ahrens[2] · Valentin Richter-Trummer[2] · Marcel Todtermuschke[2] · Franziska Bocklisch[3]

Angenommen: 23. September 2021 / Online publiziert: 10. November 2021

Zusammenfassung

Der Erfahrungsaustausch über gute Praxis ist eine Grundvoraussetzung für das Verständnis zwischen Mensch und Technik, welches dabei hilft, dem Menschen als strategischen Entscheider und Nutzer eine optimale Unterstützung zukommen zu lassen. Um der Frage nachzugehen, was gute Praxis im Bereich der Roboterprogrammierung auszeichnet, haben die Autoren eine qualitative und deskriptive Studie mit einem Vergleich zwischen $n = 2$ Versuchspersonen durchgeführt. Ein Experte und ein Novize bearbeiteten dieselbe Aufgabe: das Programmieren einer Roboterbahn zum Fräsen eines Werkstückes. Mittels Eye-Tracking Analysen und videounterstützter retrospektiver Think-Aloud Interviews wurden die Vorgehensweisen beider Probanden extrahiert und das Vorgehen anschließend formalisiert beschrieben. Zusätzlich ist ein qualitativer Vergleich zwischen den Endergebnissen gezogen worden, bei dem der Experte erwartungsgemäß besser als der Novize abschnitt. Auf Basis der Think-Aloud Protokolle wurden kognitive Prozesse identifiziert, die in diesem Kontext von besonderem Interesse sein könnten. Weiterhin wurden Augenbewegungscharakteristiken zur näheren Beschreibung einiger der kognitiven Prozesse dargelegt.

Praktische Relevanz: Die vorliegende Arbeit zeigt ein methodisches Vorgehen zur formalisierten Beschreibung und Abbildung von menschlichem Expertenwissen beim Teach-in von Industrierobotern. Sie bildet die Basis für verschiedene zukünftige Projekte, z. B. das Erstellen von Richtlinien zum praktischen und effizienten Einlernen von Anfängern oder das Umsetzen von nutzerspezifischer Assistenz im Bereich des Teach-ins von Industrierobotern.

Schlüsselwörter Wissensrepräsentation · Expertenwissen · Teach-in · Industrieroboter · Eye-Tracking

Daniel A. Darnstaedt
daniel.darnstaedt@psych.uni-halle.de

✉ Franziska Bocklisch
franziska.bocklisch@psychologie.tu-chemnitz.de

[1] Institut für Psychologie, Allgemeine Psychologie, Martin-Luther-Universität, 06108 Halle (Saale), Deutschland

[2] Fraunhofer-Institut für Werkzeugmaschinen und Umformtechnik IWU, 09126 Chemnitz, Deutschland

[3] Institut für Psychologie, Allgemeine Psychologie I & Human Factors, TU Chemnitz, 09107 Chemnitz, Deutschland

Procedure for describing human expert knowledge and cognitive processes during the teach-in of industrial robots

Abstract
The exchange of experiences about good practice is a basic requirement for understanding between human and technology, which helps to provide humans as strategic decision-makers and users with optimal support. In order to investigate the question of what characterizes good practice in the field of teach-in of industrial robots, the authors conducted a qualitative and descriptive study with the comparison of $n = 2$ participants. An expert and a novice worked on the same task, programming a trajectory for milling a workpiece. The procedures of both subjects were extracted using eye tracking analyzes and video-supported retrospective think aloud interviews and the procedures were then described in a formalized manner. In addition, a qualitative comparison was made between the final results, in which the expert, as expected, did better than the novice. Based on the think aloud protocols, cognitive processes were identified that could be of particular interest in this context. Furthermore, eye movement characteristics are reported for a more detailed description of some of the reported cognitive processes.
Practical relevance: The present work shows a methodical approach to the formalized description of human expert knowledge during the teach-in of industrial robots. It forms the basis for different future projects, e.g. the development of guidelines for handy and efficient training of beginners or the establishment of user-specific assistance in the field of teach-in of industrial robots.

Keywords Knowledge representation · Expert knowledge · Teach-in · Industrial robot · Eye-Tracking

1 Einleitung

„Deutschland als Vorreiter im Bereich der Industrie 4.0“. Das ist die Vision von Kagermann und Kollegen, als sie den Begriff „Industrie 4.0“ auf der *Hannover Messe* 2011 öffentlich einführten (Kagermann et al. 2011). Noch heute herrscht allerdings kein einheitlicher Konsens über die Definition. Unter den verschiedenen und teilweise divergenten Ansätzen findet sich jedoch ein wesentlicher Kernaspekt: die intelligente Vernetzung von Maschinen, Menschen, Abläufen und Daten (z. B. DIN 2020; Kagermann et al. 2013; Plattform-I40 2020). Ziel ist es, mittels komplexer, dezentral organisierter Strukturen die uneingeschränkte Teilhabe aller Industrieakteure zu ermöglichen, was letztlich zahlreiche Vorteile mit sich bringt (Plattform-I40 2019): flexible Produktion zur besseren Abstimmung, wandelbare Fabriken, kundenzentrierte Lösungen, optimierte Logistik oder ressourcenschonende Kreislaufwirtschaft.

Für das Zusammenführen der realen und der virtuellen Welt sind mehrere Schlüsselkomponenten notwendig (Hermann et al. 2016). Grundlage bietet (1) das Internet der Dinge (*Internet of Things*, IoT), welches den einzelnen Teilsystemen (z. B. Sensoren, Aktuatoren, Smartphones) ermöglicht, miteinander zu kommunizieren und zu interagieren, was gewährleistet, dass ein gemeinsames Ziel im (Produktions-)Ablauf verfolgt werden kann. Des Weiteren kommen (2) Cyber-Physische Systeme (CPS) zum Einsatz, die der Verwaltung der verbundenen Systeme dienen. Hier werden physische Prozesse und Ressourcen mit der Rechenfähigkeit von Computern integriert. Aus der physischen Welt werden Daten in Echtzeit in den *Cyber Space* übermittelt und dort mittels geeigneten Datenmanagements und entsprechender Analytik bewertet (Lee et al. 2015). Die Ergebnisse werden wieder zurück an die physischen Systeme gesendet und daraus resultierende Feedbackschleifen ermöglichen es Maschinen, Eigenschaften der Intelligenz, Resilienz und Selbst-Adaptierung zu entwickeln. Die Kombination der beiden ersten Schlüsselkomponenten IoT und CPS vereint (3) die *Smart Factory*: über die von den Netzwerken und Interaktionen generierten Informationen kann allen beteiligten Akteuren kontextbezogene Assistenz geboten werden, um die Produktion zu optimieren und das Arbeitsgeschehen reibungslos zu gestalten. Es kommen laut Hermann et al. (2016) bei der Betrachtung von Szenarien der Industrie 4.0 vier Gestaltungprinzipien zum Einsatz. Die beiden oben bereits angesprochenen Punkte Vernetzung und Informationstransparenz bilden die Basis für dezentrale Entscheidungen, die aufgrund der Zusammenführung lokaler und globaler Informationen getroffen werden können und welche die Beteiligten in die Lage versetzen, weitestgehend autonom zu handeln. Die Rolle des Menschen verschiebt sich deshalb immer mehr vom Operateur hin zum strategischen Entscheider und flexiblen Problemlöser. Dabei soll ihm im Sinne der *Smart Factory* technische Assistenz gewährt werden. Diese kann virtuell ausfallen, d. h. relevante Informationen werden für ihn zusammengetragen und entsprechend visualisiert, oder von physischer Natur sein, z. B. durch den Einsatz verschiedener Industrieroboter, welche sich möglichst effizient und harmonisch bewegen sollten, damit ein reibungsloser und sicherer Arbeitsablauf gewahrt bleibt.

Ähnliche Gestaltungsfragen diskutieren auch Botthof und Hartmann (2015) und plädieren für mehr dezentrale Entscheidungen. Weiterhin sehen die Autoren die Rolle des Menschen ebenfalls weniger als Bediener der Maschinen denn als Nutzer und die Technik solle den Menschen nicht ersetzen, sondern vielmehr als Unterstützung dienen. Dem Menschen kommt demnach weiterhin eine wichtige Rolle im Industriegeschehen zu. Dabei wird in Zukunft verstärkt darauf zu achten sein, dass auf Veränderungen in der Interaktion zwischen Mensch und Technik reagiert wird, die im Prozess der Informatisierung der Arbeitswelt noch enger zusammenrücken werden. Diese Veränderungen werden zunächst vorrangig in den Bereichen der Kommunikation und Interaktion stattfinden. Dabei ist es essenziell, dass der Mensch die komplexer werdenden technischen Systeme weiterhin versteht und diese ausreichend transparent gestaltet sind. Umgekehrt sollte die Technik aber auch (Teil-)Aspekte menschlicher Kognitionen und Handlungsweisen sowie seine Absichten erfassen können, um zielgerichtete (adaptive) Assistenz zu realisieren. Laut Botthof und Hartmann sind dabei neue Qualifizierungsbedürfnisse und eine neue Arbeitsorganisation von besonderer Bedeutung, was eine innovative Aus- und Weiterbildung zur Folge hat, wobei ein Erfahrungsaustausch über gute Praxis ein wichtiger Bestandteil ist. Genau hier setzt die vorliegende Arbeit an.

Die Programmierung von Industrierobotern erfolgt je nach Datenlage entweder mit Hilfe sogenannter Offline-Programmier-Software – sofern alle relevanten Daten der Bauteile, Fertigungsmittel und evtl. Störkonturen wie weitere Maschinen oder Werkstattinformationen als digitale dreidimensionale Geometrie vorliegen – oder direkt vor Ort im sogenannten Teach-in Verfahren durch einen Roboterprogrammierer, der unmittelbar auf alle Randbedingungen des Fertigungsprozesses eingehen kann. Insbesondere im industriellen Umfeld von Klein- und mittelständischen Unternehmen (KmU) besitzt die Teach-in Programmierung größere Bedeutung.

Industrieroboter entlasten den Menschen ergonomisch beim Handhaben großer oder schwerer Bauteile, werden aber insbesondere auch zunehmend für wertschöpfende Fertigungsprozesse wie Fugen, Umformen, Lackieren oder Zerspanen eingesetzt. Während beim einfachen Handhaben von Bauteilen eine möglichst kurze und schnelle Bahn zwischen einem definierten Start- und Endpunkt (unter Berücksichtigung von eventuellen Störkonturen) zu definieren ist, liegt bei den genannten Fertigungsprozessen der Schwerpunkt der Programmierung insbesondere in der Realisierung einer technologisch geeigneten Bahn, was deutlich höhere Ansprüche an die Umsetzung der Programmierung stellt. Zudem gibt es oftmals nicht nur genau eine richtige Lösung, sondern unterschiedliche – hinsichtlich verschiedener Kriterien paretooptimale – Bahnen, deren Auswahl und Anpassung von den Erfahrungen des Roboterprogrammierers abhängen. Je nach Fertigungsaufgabe werden die programmierten Bahnen dann unterschiedlich häufig genutzt, bevor eine andere Aufgabe des Roboters programmiert wird, was sich insbesondere bei solchen Roboterbahnen für kleine Stückzahlen als relevanter wirtschaftlicher Faktor für den Gesamtnutzen des Roboters darstellt.

Was macht gute Praxis in der Roboterprogrammierung im industriellen Kontext aus? Wie kann das erfasst werden?

In Abschn. 2 ist ausgeführt, was Expertenwissen in diesem Applikationsbeispiel ausmacht und welche kognitiven Prozesse dabei eine Rolle spielen könnten. Abschn. 3 behandelt die im Zuge dieser Arbeit genutzte methodische Vorgehensweise. Die resultierenden Ergebnisse aus der Untersuchung sind in Abschn. 4 präsentiert. Abschließend werden Implikationen für die Zukunft in Abschn. 5 diskutiert.

2 Theoretischer Hintergrund

Cyber-Physische Systeme (u. a. Lee 2008; Lee et al. 2015) verknüpfen die physische mit der virtuellen Welt mit dem Ziel, den Autonomiegrad des technischen Systems zu erhöhen (vgl. Plattform-I40 2019b). Hierzu werden durch (smarte) Sensornetzwerke große Datenmengen über Zustand und Leistungsfähigkeit des technischen Systems und seiner Teilkomponenten aufgezeichnet und ausgewertet (z. B. mittels geeigneter Algorithmen der Künstlichen Intelligenz, KI). Im *Cyber Space* soll dann ein virtuelles Modell des technischen Systems abgebildet werden, das die Realität so gut charakterisiert, dass eine Art „maschinelle Selbstwahrnehmung“ entsteht (z. B. Informationen zur Zustandshistorie, Selbstvergleich, Verhaltensvorhersagen). Dies bildet die Grundlage, den Feedbackkreis zu schließen und das reale technische System durch das virtuelle Modell steuern zu können (vgl. Bild 3 in Bocklisch et al. 2020). Mittlerweile gibt es mehrere Ansätze, um die Rolle des Menschen in CPS zu diskutieren bzw. diesen einzubeziehen (Madni 2019: Adaptive Cyber-Physical-Human Systems; Wang 2010: Cyber-Physical-Social Systems; Nelles et al. 2016). Bocklisch et al. (2020) diskutieren und ergänzen die CPS-Konzeption in dreierlei Hinsicht um die menschliche Perspektive: (1) Da der Mensch aufgrund seiner Rolle als kreativer Problemlöser und Wissens-/Erfahrungsträger nicht vollständig zu ersetzen ist, wird weiterhin eine Schnittstelle benötigt, bei der allerdings stärker Assistenz- als Bedienfunktionalitäten im Vordergrund stehen werden. Über diese kann der Nutzer auf gut organisierte und präsentierte Informationen über den aktuellen Zustand des technischen Systems zugreifen, um Unterstützung für seine Entscheidungen zu erhalten und somit seine Arbeitsaufgabe besser erfüllen zu können.

(2) Weiterhin regen die Autoren im Sinne einer kognitionsbasierten Mensch-Technik-Interaktion an, die für die Arbeitsaufgabe relevanten kognitiven Prozesse des Menschen über daten- und expertenbasierte Ansätze möglichst auch explizit in der virtuellen Welt abzubilden (z. B. kognitive Modelle). Mittels einer übereinstimmenden KI-Modellierungsarchitektur oder durch Verschränkung verschiedener geeigneter KI-Verfahren kann eine Verbindung zwischen dem kognitiven Modell und dem virtuellen Modell des technischen Systems geknüpft werden. Wenn auf diese Weise die menschliche Perspektive expliziter als bisher erfasst und modelliert wird, kann ein konzertierteres Zusammenspiel von Mensch und Technik ermöglicht werden. In Bocklisch et al. (im Druck) wird die Integration menschlicher Kognition in CPS am Beispiel der Fertigungstechnik „Thermisches Beschichten" exemplarisch mit einem mehrdimensionalen Fuzzy Pattern Classification KI Modell gezeigt. Abschließend (3) wird die Perspektive des Menschen als wesentlicher Akteur bei der Konzeption und Realisierung technischer Systeme im physischen und virtuellen Raum hervorgehoben, die im Sinne eines ganzheitlichen Systemansatzes zu enger transdisziplinärer Zusammenarbeit von Human-/Sozial- und Technikwissenschaften sowie Industrieakteuren auffordert (Madni 2018). Die Hyperkonnektivität und wachsende Komplexität von CPS in der Industrie kann sonst nicht ausreichend kontrolliert werden. Ausgehend von den ersten beiden Punkten wird die Aufgabe des Teach-ins von Industrierobotern herausgegriffen, um hierfür exemplarisch eine formalisierte Beschreibung des menschlichen Expertenwissens zu generieren. Auf diese kann bei der Weiterentwicklung von Assistenzsystemen in diesem Bereich zurückgegriffen werden. Da beim Teach-in ein hohes Maß an Erfahrung und Expertenwissen nötig sind, werden die psychologischen Grundlagen hierzu kurz skizziert.

2.1 Expertise

Der Begriff der Expertise ist breit gefächert und nicht einheitlich definiert. Nach Posner (1988) ist ein Experte eine Person, die dauerhaft hervorragende Leistung in einer Domäne erbringt. Krems (1994) beschreibt Expertise noch etwas konkreter als die bereichs- und aufgabenspezifische Problemlösefertigkeit in einem Sachgebiet, die in die Lage versetzt, dauerhaft Hervorragendes zu leisten. Hier soll Expertise unter folgenden Gesichtspunkten verstanden werden: (1) Expertenwissen wird immer nur in einem bestimmten Bereich erworben und hängt stark mit den entsprechenden Domänencharakteristika zusammen, d. h. es ist gar nicht oder nur sehr eingeschränkt auf andere Bereiche generalisierbar. (2) Expertise kann nur über ausgiebige Erfahrung und Übung erlangt werden. Hier wird öfter von einem Zeitraum von 8–10 Jahren gesprochen (z. B. Krems 1994; Parnin et al. 2017). (3) Ein Experte hat auf seinem Gebiet einen gewissen messbaren Performanz- bzw. Effizienzvorteil gegenüber nicht so Erfahrenen. Das kann sich u. a. durch die Bearbeitung von mehr Aufgaben mit weniger Kosten (z. B. Zeit, Aufwand, Fehlerquote) auszeichnen. (4) Dies kann erreicht werden, da der Erwerb von Expertise mit einer Veränderung der Wissensstrukturen der betreffenden Person einhergeht. Informationen werden in verbundenen, bedeutungsvollen Einheiten, sogenannten *Chunks*, aufgenommen und verarbeitet (z. B. Miller 1956; Chase und Simon 1973). Durch Erfahrung auf einem bestimmten Gebiet können die Chunks zu größeren Einheiten verbunden werden. Die entstehenden *Templates* (Gobet und Simon 1996) sind in einer Art Netzwerk mit zunehmend hierarchischer Ordnung aufgebaut (Chi et al. 1981). Dadurch können domänenspezifische Muster schneller kategorisiert, Suchverhalten effizienter gestaltet und Probleme schneller gelöst werden.

2.2 Expertise beim Programmieren

Auch beim Programmieren als eine Domäne, in der Expertise entwickelt werden kann, spielt das Bilden von *Chunks* bzw. *Templates* eine wichtige Rolle. Es trägt dazu bei, die Fehleranzahl zu minimieren und die Bearbeitungszeit zu verkürzen (Vessey 1985). Ein Experte zeichnet sich im Allgemeinen dadurch aus, dass die Größe der gebildeten *Chunks* zunehmen und somit mehr Informationen integriert werden, weniger durch eine erhöhte Gesamtanzahl von *Chunks* (Gobet und Clarkson 2004). Diese strukturellen Unterschiede konnten Ye und Salvendy (1994) auch für programmierspezifische Aufgaben aufzeigen. Daraus folgt, dass Programmier-Experten mehr programmierrelevante Informationen im Arbeitsgedächtnis zugänglich halten, die für die Erledigung der Arbeitsaufgabe genutzt werden können.

Das zielgeleitete Erstellen eines Programmcodes ist nach Davies (1993) ein inkrementeller Problemlöseprozess, im Verlaufe dessen feste Problemlöseepisoden und eine fortwährende Problemneubewertung die zu verwendende Strategie bestimmen und die Angemessenheit von gefundenen Lösungen ständig überprüft wird. Das Ansammeln und Speichern von Faktenwissen (deklaratives Wissen, z. B. über Syntax und Semantik einer Programmiersprache) ist dabei nicht die einzige Determinante. Ebenso spielt eine sehr große Rolle, wie das Wissen organisiert ist und nach welchen Kriterien Programmierstrategien ausgewählt, angewendet und im Verlauf des Programmierens verändert werden. (z. B. Davies 1993; Kasto 2012; Parnin et al. 2017). Unterschiede zwischen Experten und Novizen in der generellen Verwendung von Strategien beim Programmieren wurden bereits vielfach untersucht. Nach Parnin et al. (2017) ist das Vorgehen beim Programmieren stark von der jeweiligen Programmiersprache und -umgebung abhängig

Tab. 1 Liste der kognitiven Prozesse (kP), die beim Programmieren in der Sprache „C“ vorkommen; Unterteilt in fünf Kategorien: Sinneseindrücke, unterbewusste kP, metakognitive kP, höhere kP und zusätzliche kP. (Übersetzt, vgl. Renumol et al. 2010)
Table 1 List of cognitive processes (CP) that occur when programming in the “C” language; divided into five categories: sensational CP, subconscious CP, metacognitive CP, higher CP and additional CP. (translated, see Renumol et al. 2010)

Sinneseindrücke	Unterbewusste kP	Meta-kP	Höhere kP	Zusätzliche kP
Sehen	Handlungen, Emotionen, Zielsetzung, Gedächtnis, Motivation, Selbstbewusstsein, Bereitschaft	Abstrahieren, Aufmerksamkeit, Kategorisierung, Konzeptetablierung, Wissensrepräsentation, Auswendigkernen, Suche	Analogie, Analysieren, Verständnis, Kreieren, Entscheidungsfindung, Deduktion, Erklären, Verbildlichen, Induktion, Lernen, Planen, Problemlösen, Quantifizieren, logisches Denken, Erkennen, Synthese	Hypothesen bilden, Abfragen, Iterieren, Überwachen, Erinnern, Wiederholen, Übersetzen

und man könne wenig pauschale Aussagen über Unterschiede treffen. Sie untersuchten vor allem das Verständnis von vorliegenden Programmen und stellten mögliche voneinander zu unterscheidende Ansätze dar. (1) *Top-down* Strategien gehen von einer Starthypothese aus, welche anschließend schrittweise verfeinert und spezifiziert wird. Bei einem *bottom-up* geleiteten Prozess hingegen werden Details des Codes so lange zu semantischen Chunks verbunden, bis ein tiefgreifendes Verständnis aufgebaut worden ist. (2) Das Vorgehen kann im Voraus strategisch geplant sein oder aber eher opportunistisch erfolgen, d. h. an den jeweiligen Bedarf angepasst werden. (3) Die Methodik kann struktur- oder variablenbasiert aufgebaut sein. (4) Oder aber es werden ganz spezifische Strategien für ebenso spezifische Probleme entwickelt, die nicht weiter verallgemeinert werden können.

Anderson (2007) geht davon aus, dass beim Programmieren hauptsächlich *top-down* Strategien verwendet werden, da i. d. R. lediglich das Ziel vorgegeben sei, was eine hypothesengeleitete Vorgehensweise begünstige. Novizen würden dabei eher zuerst in die Tiefe gehen und das erste Teilproblem bis auf die unterteste Ebene aufschlüsseln wollen, während Experten zunächst die gesamte Breite der ersten Ebene der Planungshierarchie ausbauen wollten, da somit einzelne voneinander abhängige Teile der Aufgabe entsprechende Zusammenhänge besser erkennen ließen. Experten profitieren nach einer Studie von Peitek et al. (2020) beim Verständnis von Programmcodes mehr als Novizen von einer Programmstruktur, die *top-down* Strategien fördert. Vessey (1985) hingegen argumentiert in einer anderen Richtung: Novizen nutzten überwiegend formale Modelle, mit denen sie möglichst situationsunabhängig an das Ziel gelangen könnten. Experten aber agierten vermehrt reaktiv, ließen sich also datengetrieben eher von den äußeren Umständen beeinflussen und passten ihr Vorgehen dahingehend an.

In dieser Arbeit wurde eine spezifische Art des Programmierens von Industrierobotern und damit zusammenhängende Besonderheiten untersucht, welche sich in bestimmten Aspekten von anderen Programmierumgebungen unterscheidet. Diese sind im Methodenteil näher beschrieben.

2.3 Kognitive Prozesse

Der Rolle des Menschen in der Industrie kommt und wird auch in Zukunft eine zentrale Bedeutung zukommen, sowohl beim Programmieren und Einrichten von Robotern als auch in der Interaktion mit diesen. Die Grundlage für die Suche nach kognitiven Prozessen, die beim Programmieren von Industrierobotern eine Rolle spielen, könnte das Informationsverarbeitungsmodell von Wickens et al. (1998) darstellen. In dieses lassen sich viele kognitive Prozesse einordnen, die in Beziehung zu Wahrnehmung, Enkodierung von Informationen, zentralen Verarbeitungsprozessen (Aufmerksamkeit, Gedächtnis, Entscheidungsfindung etc.) sowie der Auswahl und Ausführung von Antworten bzw. Reaktionen stehen. Welche davon sind jedoch relevant beim Programmieren?

Dieser Frage widmen sich Renumol et al. (2010) und stellen eine Liste von 42 kognitiven Prozessen zusammen (Tab. 1). Als Ausgangspunkt nutzen sie das *Layered Reference Model of The Brain* von Wang et al. (2003) und *Bloom's Taxonomy* (Bloom 1956), welche zunächst kognitive Prozesse im Allgemeinen definieren, und ergänzen diese um acht weitere, die speziell im Bereich der Programmierung mit *C* von Relevanz sind. Sie definieren kognitive Prozesse als Prozesse im Gehirn, mit denen Informationen verarbeitet und/oder abgerufen und/oder gespeichert werden können. Ein kognitiver Prozess kann andere auslösen und/oder den Geisteszustand einer Person verändern. Die Autoren stellen weiterhin fest, dass das Programmieren ein Zusammenspiel aus niederen (z. B. Aufmerksamkeit, Gedächtnis) und höheren kognitiven Prozessen (z. B. Entscheidungstreffen, Problemlösen) erfordert sowie verschiedene kognitive Fähigkeiten voraussetzt, was es zu einem komplexen und schwer zu erlernenden Prozess macht. Die Generalisierbarkeit dieser Studie ist begrenzt, zum einen weil es nur wenige Versuchspersonen sind, die untersucht wurden. Zum anderen erfasst die Liste kognitive Prozesse, die speziell beim Programmieren mit der Programmiersprache *C* eine Rolle spielen. Da das Programmieren eines Roboters in einer anderen Umgebung jedoch davon abweichen könnte, bleibt die Frage offen, welche dieser Prozesse auch in dieser Arbeit relevant sind.

2.4 Fehleranalyse

Ein weiterer Gesichtspunkt für den Vergleich zwischen Experte und Novize sind Fehler, die während des Teach-ins eines Roboters auftreten können. Diese lassen sich bspw. nach Reason (1990) in drei Ausführungsebenen unterteilen. Auf der (1) fähigkeitsbasierten Ebene treten Schnitzer (Gedächtnisfehler) und Patzer (Aufmerksamkeitsfehler) auf, die hauptsächlich durch automatische Prozessoren wie Schemata kontrolliert werden. Diese geschehen zumeist bei Routinehandlungen und die Aufmerksamkeit liegt häufig auf etwas anderem als der augenblicklichen Aufgabe. Die Kontrolle auf dieser Ebene wird von den automatischen Prozessoren (beim Programmieren vorrangig von Regeln) durchgeführt und ist hauptsächlich von intrinsischen Faktoren beeinflusst. (2) Regelbasierte und (3) wissensbasierte Fehlleistungen hingegen treten während des Problemlöseprozesses auf, wobei die Aufmerksamkeit in der Regel bei problembezogenen Fragen liegt. Hier sind es vor allem beschränkte, bewusste Prozesse, die der Kontrolle dienen und größtenteils extrinsische Faktoren mit Einfluss.

3 Methodik

Aus den vorangehenden Überlegungen und theoretischen Darstellungen ergeben sich verschiedene Fragestellungen. Ausgehend von der Bearbeitung einer einfachen Aufgabe zum Teach-in eines Industrieroboters wird das dabei verwendete Vorgehen formalisiert dargestellt, Unterschiede zwischen Experte und Novize beschrieben, kognitive Prozesse betrachtet und Augenbewegungscharakteristiken untersucht (Götz 2020).

3.1 Versuchsteilnehmer

Die beiden ProbandInnen dieser Studie sind MitarbeiterInnen des *Fraunhofer-Instituts für Werkzeugmaschinen und Umformtechnik Chemnitz IWU* (1 weiblich, 1 männlich im Alter von 30 und 33 Jahren) und wurden entsprechend ihrer Erfahrung vom Gruppenleiter ausgewählt. Die Einteilung in Experte und Novize wurde anhand eines Selbsteinschätzungsfragebogens zur Messung von Programmiererfahrung (siehe Feigenspan et al. 2012) verifiziert. Der Experte schätzte sich in allen wesentlichen Punkten besser ein als der Novize: allgemeine Erfahrung beim Programmieren, Vergleich zu Kollegen, Vergleich zu einem hypothetischen Experten, Erfahrung mit der hier angewandten Software. Experte und Novize unterschieden sich in ihrem Ausbildungshintergrund. Während der Experte aus dem Bereich der Elektroinstallation (SPS-Fachkraft) kam, hatte der Novize einen Masterabschluss in Mathematik. Beide Mitarbeiter tragen im Alltag eine Brille zur Korrektur der Sehfähigkeit und wichen für den Versuch auf Kontaktlinsen aus, um die Messung der Blickbewegungen mittels Eye-Tracker zu ermöglichen.

3.2 Material und Geräte

3.2.1 Aufzeichnung

Während der Aufgabenbearbeitung wurden die Augenbewegungen der Probanden mittels des portablen Eye-Trackers „SMI ETG“ aufgezeichnet. Über den in einem Android-Gerät installierten *Smart Recorder* wurden die Augen mit einer Rate von 60 Hz binokular und das Szenenbild mit 30 Hz aufgenommen.

Die anschließenden retrospektiven *Think-Aloud* Interviews (vgl. Van Gog et al. 2005; Ruckpaul et al. 2015; Van Den Haak et al. 2003; Whalley und Kasto 2014) wurden aufgezeichnet und analysiert. Sie bilden die Grundlage für die später formalisierte Beschreibung der Vorgehensweisen. Die beiden Befragten sahen nach Abschluss der Programmieraufgabe die Videoaufzeichnung der Eye-Tracking Kamera und bekamen die Anweisung, Gedankengänge und Intentionen hinter ihren Handlungsschritten zu schildern. Währenddessen haben die Interviewer gezielt Fragen gestellt, um auf spezifische Schritte näher einzugehen.

3.2.2 Industrieroboter und Software

Bei dem einzurichtenden Roboter handelte es sich um einen „KR 150 R2700 extra“ der Firma *KUKA* (KUKA 2020b). Verwendet wurde das standardmäßige Betriebssystem von *KUKA*, die „KUKA.SystemSoftware“ (KSS) in der Version 8.3.17. Diese läuft auf Windows und beinhaltet alle Grundfunktionen wie bspw. die Bahnplanung oder das I/O-Management und auch schon einige erweiterte Funktionalitäten (KUKA 2020a). Bedient wird der Roboter bzw. die Software zumeist über das „KUKA smartPAD“ (*Touch-Panel*). Ähnlich wie in anderen Programmiersprachen können Befehle direkt eingegeben werden. Zusätzlich aber gibt es verschiedenste Auswahl- und Einstellmöglichkeiten in einer grafischen Nutzeroberfläche, die das vereinfachte Implementieren bestimmter Funktionen erlauben. Des Weiteren befindet sich an der Seite eine 6D-Maus, über die ein manuelles Bewegen der Roboterachsen in einem frei wählbaren Koordinatensystem realisiert wird, was ein weitgehend intuitives Teach-in erlaubt. Der Programmierer kann als Instruktor die Bewegung des Roboters direkt steuern, verschiedene Punkte anfahren und speichern, um so einen kompletten Arbeitszyklus zu erschaffen, den der Roboter später im Automatikmodus selbstständig ausführen kann. Das Panel bietet den Vorteil, dass der Programmierer sich während des Bedienens und Einrichtens des Roboters flexibel in der Roboterzelle bewegen kann. Das ermöglicht ihm

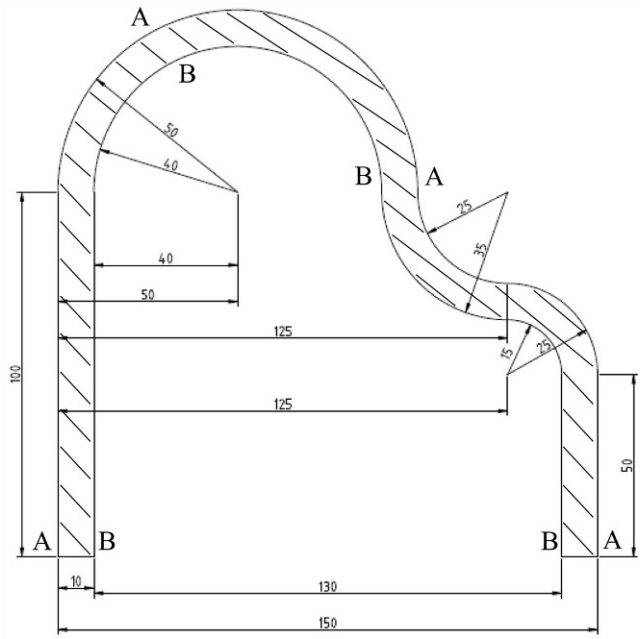

Abb. 1 Technische Zeichnung des Stegs (*schraffierte Fläche*) aus der Aufgabenstellung, welcher durch die beiden eingefrästen *Linien A und B* begrenzt wird. Die Zeichnung diente den Programmierern später auch als Schablone beim Bearbeiten

Fig. 1 Technical drawing of the bar (*hatched area*), which is delimited by the two *milled lines A and B*. The programmer later also used the drawing as a template for editing

aufgrund der visuellen Kontrolle ein genaues Teachen von Punkten. Der Roboter wurde im Voraus mit einer Frässpindel ausgestattet, um die nachfolgend beschriebene Aufgabe bearbeiten zu können.

3.2.3 Versuchsaufbau und Aufgabenstellung

Zu Beginn der Erhebung wurde den beiden Probanden gleichzeitig die Aufgabenstellung erklärt und kurz besprochen. Sie erhielten eine technische Zeichnung (Abb. 1) und eine mündliche Beschreibung der Aufgabe. Ziel der Aufgabe war es, auf einer flachen Holzplatte einen Steg mit zwei parallelen Wänden herzustellen (schraffierte Fläche in Abb. 1), d. h. im inneren und im äußeren Bereich jeweils einen Graben einzufräsen (durchgezogene Linien A und B in Abb. 1). Dafür mussten zwei separate Teilaufgaben durchgeführt werden: (1) die Außenkante des Stegs (Linie A) sollte direkt auf der Linie eingefräste werden und für (2) die Innenkante (Linie B) musste die Frässpitze nicht direkt auf der Linie verlaufen, sondern unmittelbar daneben, um eine Breite von 10 mm für den Steg zu gewährleisten. Das Vorgehen zum Erreichen des Ziels wurde den Probanden offengehalten, lediglich die Geometrie des Werkstücks am Ende war ausschlaggebend für die Bewertung. Für die Bearbeitung bekamen die Versuchspersonen eine Zeitvorgabe von 30 min, welche von den Beteiligten als recht knapp eingestuft worden ist.

Der Experte und der Novize bearbeiteten dann die Aufgabe nacheinander, und ohne gegenseitige Anwesenheit, in derselben Roboterzelle. Nach der Kalibration des Eye-Trackers (3-Punkt-Kalibration) wurde die Zeit gestoppt und die jeweilige Versuchsperson begann mit der Aufgabenbearbeitung. Eine Woche später führte die Versuchsleitung getrennt für Experte und Novize Interviews (videogestütztes *Think-Aloud*) durch, in deren Verlauf die beiden Befragten ihre Gedanken und Intentionen wie oben bereits näher beschrieben schilderten. Die Gespräche fanden zu diesem Zeitpunkt aufgrund von Beschränkungen von persönlichen Kontakten digital über Webtelefonie statt. Damit waren alle Daten erhoben und es konnte mit der Auswertung begonnen werden.

4 Ergebnisse

4.1 Formalisierung des Arbeitsprozesses

Zur Formalisierung des Vorgangs beim Einlernen eines Industrieroboters wurden die Audiodateien der Interviews zunächst verschriftlicht und die Transkripte anschließend jedes für sich in einer der *Grounded Theory Analysis* (Charmaz und Belgrave 2012) angelehnten Vorgehensweise bearbeitet. Als erstes wurde die Wort-für-Wort-Darlegung initial kodiert, d. h. es wurden Markierungen gesetzt und Bemerkungen (Label) notiert. Eine stichpunktartige Zusammenfassung der Abläufe des Programmierens stellte die Basis für die weitere Untersuchung dar, indem die wichtigsten Label, die zur eindeutigen Beschreibung des Prozesses nötig sind, ausgewählt wurden. Anschließend erfolgte eine Zusammenfassung der einzelnen Handlungen in übergeordnete Schritte, welche wiederum in große Abschnitte eingeteilt werden konnten. Das Ergebnis sind zwei verschiedene Vorgehensweisen zum Bearbeiten der Aufgabe, die als *Flow-Charts* in Abb. 2 (Experte) und 3 (Novize) dargestellt sind. Die Grafiken veranschaulichen eine verallgemeinerte Vorgehensweise wie sie die Probanden im Kern zeigten. Experte und Novize sind jedoch bei der Aufgabenbearbeitung ab und an von diesen idealisierten Schemata abgewichen, haben kleinere Teilschritte mal vertauscht oder ausgelassen. Im Folgenden sollen die beiden allgemeinen Vorgehensweisen dargestellt werden (Abb. 2 und 3).

Trotz der unterschiedlichen Herangehensweise bei der Programmierung der Roboterbewegung ist den Abb. 2 und 3 zu entnehmen, dass die grundsätzliche Bearbeitung der Aufgabe sehr ähnlich abgelaufen ist. Beide Schemata können in sechs große Abschnitte unterteilt werden, die sich wiederum in zwei separate Blöcke einteilen lassen. Sowohl Novize als auch Experte trafen zunächst Vorbe-

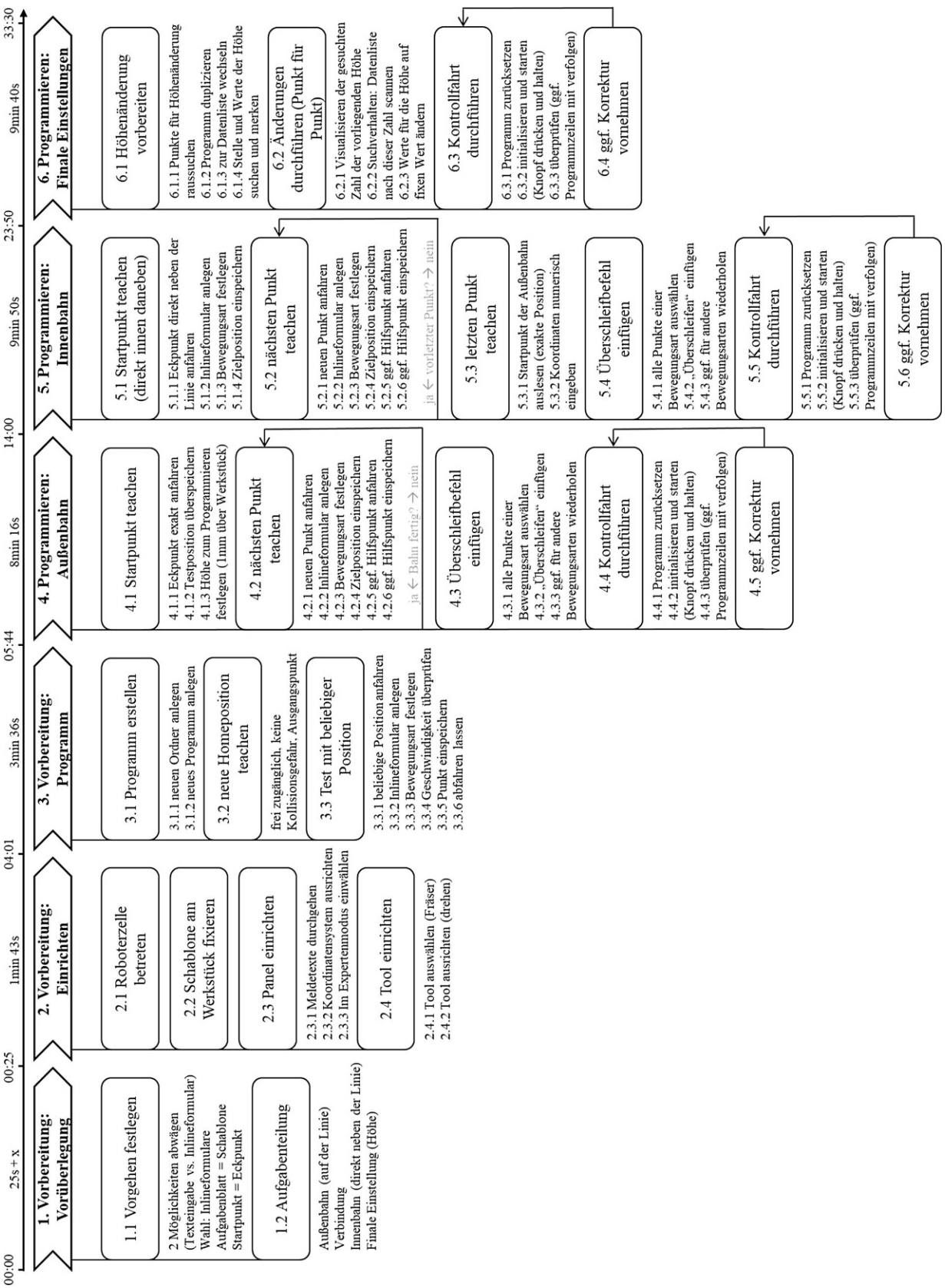

Abb. 2 *Flow Chart* des Ablaufes beim Teach-in mit der Aufgabe, einen Steg in ein Werkstück zu fräsen (Experte)

Fig. 2 Flow chart of the process during teach-in with the task of milling a bar in a workpiece (expert)

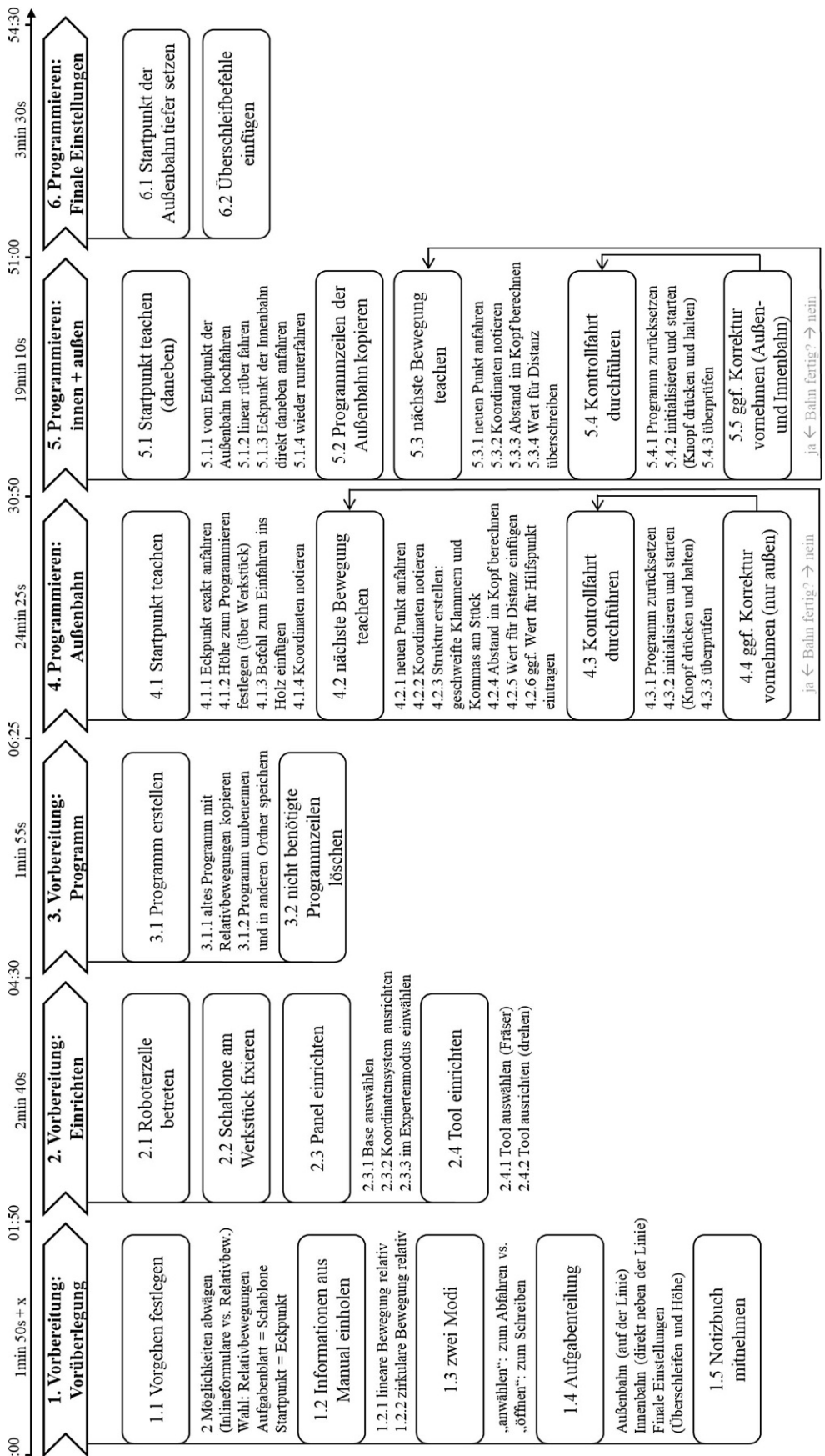

Abb. 3 *Flow Chart* des Ablaufes beim Teach-in mit der Aufgabe, einen Steg in ein Werkstück zu fräsen (Novize)

Fig. 3 Flow chart of the process during teach-in with the task of milling a bar in a workpiece (novice)

reitungen (Block 1), bevor sie mit der eigentlichen Programmierung (Block 2) begannen. Beide legten zunächst ihr Vorgehen fest, wobei sie jeweils zwischen zwei Alternativen abwogen. Für den Experten stand fest, dass er die einzelnen Punkte der Trajektorie direkt anfahren und einspeichern wird. Er nutzte von der KUKA.SystemSoftware vorgegebene *Inlineformulare*, welche ihm die Struktur der Bewegungsabläufe vorgaben, in die er dann noch die passenden Werte entsprechend der zuvor angefahrenen Punkte einspeichern musste. Der Novize hingegen entschied sich für die Nutzung von Relativbewegungen, die er als Text händisch in das Panel eingab.

Beide teilten sich die Aufgabe gedanklich in separate Teilschritte ein: das Teachen (1) der Außenbahn, (2) der Innenbahn und (3) abschließende Einstellungen zur Kompletierung des Programms. Der Experte baute noch den Zwischenschritt der Erstellung des Übergangs zwischen beiden Bahnen ein und der Novize wich später von der strengen Unterteilung ab und bearbeitete die Innenbahn dann parallel zur Außenbahn. Des Weiteren griff der Novize in dieser Phase noch auf Informationen zu seiner gewählten Vorgehensweise aus dem Handbuch von *KUKA* zurück und übertrug sich diese in ein Notizbuch, welches er während des Teach-ins neben sich in der Roboterzelle liegen hatte. Zusätzlich überlegte er sich, wie die zwei Modi („anwählen“ vs. „öffnen“) zu nutzen sind, die am Panel eingestellt werden können.

Den zweiten Schritt in Block 1 gingen Novize und Experte auf fast identische Weise an. Nach dem Eintritt in die Roboterzelle legten sie das Aufgabenblatt auf das Werkstück, welches beide als Schablone zum Anfahren der Punkte verwendeten. Dann wurde das Panel eingerichtet, indem sie die Basis auswählten, auf die sich das Koordinatensystem bezieht (in diesem Fall die Werkbank, auf der das Werkstück lag). Anschließend loggten sich die Programmierer in den Expertenmodus ein (Modus im Panel mit erweiterten Funktionen). Nur darüber konnte bspw. das Einrichten des Tools erfolgen; es musste ausgewählt werden (hier der Fräser) und die Werkzeugspitze wurde durch Drehen ausgerichtet, sodass sie normal (senkrecht) auf das Werkstück auftrifft.

Das Programm erstellten beide im dritten Schritt. Der Experte legte ein neues Programm an, wohingegen der Novize ein vorhandenes nutzte, in dem er bereits früher Relativbewegungen programmiert hatte, und entsprechend der aktuellen Fragestellung anpasste (z. B. durch Löschen nicht benötigter Programmzeilen). Der Experte führte dann zwei zusätzliche Schritte durch, die beim Novizen nicht vorkamen: (1) er legt sich eine neue Home-Position fest, welche eine zwischen Roboterwerkzeug und Werkstück kollisionsfreie Ausgangsposition etwas entfernt vom zu bearbeitenden Werkstück darstellt. Weiterhin (2) nutzte der Experte die Vorbereitungsphase, um an einem beliebig gewählten Punkt alle getroffenen Einstellungen zu testen und zu sehen, ob die Inlineformulare erwartungsgemäß funktionieren.

Damit begann Block 2, das eigentliche Programmieren. Beide legten sich zunächst einen Startpunkt der eigentlichen Bearbeitung (einen der Eckpunkte auf der technischen Zeichnung) fest sowie die Höhe, auf der die Bahn parallel zur Bauteiloberfläche programmiert werden sollte. Der Experte wählte 1 mm über der auf dem Bauteil aufgelegten Schablone, um in dieser Höhe den Roboter über die gesamte Bahn hinweg einzuteachen. Der Novize hingegen legte seinen Startpunkt vorerst 5 mm über der Bauteiloberfläche fest, fügte dann einen Befehl ein, der den Fräskopf genau diese 5 mm nach unten fahren lässt, und landete damit auch knapp über der Schablone zum weiteren Programmieren. Das Ziel dahinter war, nach dem Programmieren der Trajektorie den Startpunkt diese 5 mm in Richtung Bauteil zu verschieben, um dann die komplette Fräsbahn innerhalb des Werkstücks zu vollziehen (nicht darüber), da alle anderen Punkte relativ zum vorangehenden eingegeben wurden. Er notierte sich außerdem noch die Koordinaten des Startpunktes, da dies für sein weiteres Vorgehen notwendig war.

Im nächsten Schritt macht sich nun der generelle Unterschied zwischen den beiden Arten der Programmierung bemerkbar. Der Experte fuhr einen neuen Punkt an, erstellte ein Inlineformular, legte darin die Bewegungsart fest (für diese Aufgabe linear oder zirkular) und speicherte die Zielposition des eben angefahrenen Punktes ein. Falls es sich um eine Zirkularbewegung handelte, musste er noch einen Hilfspunkt anfahren und dessen Koordinaten in das Inlineformular einspeichern. Diesen Vorgang wiederholte er, bis die Außenbahn vollständig war. Anschließend fügte er für alle Punkte Überschleifbefehle ein. Dadurch wurden die Punkte nicht nacheinander mit jeweils einer kurzen Pause angefahren, sondern es entstand eine flüssige Bewegung, indem beim Erreichen des Überschleifbereichs des Punktes direkt zum nächsten Punkt abgebogen wird. Das Einfügen der Überschleifbefehle musste für jede Bewegungsart separat erfolgen. Erst als er mit all diesen Punkten fertig war, fuhr der Experte seine erste Kontrollfahrt. Er setzte das Programm zurück, initialisierte und startete. Während des Abfahrens achtete er darauf, dass die Frässpitze exakt auf der Linie entlangfährt und sich an der Bewegungshöhe nichts verändert. Fand er einen Fehler, korrigierte er ihn, und fuhr dann eine weitere Kontrollfahrt, bis er mit dem Resultat zufrieden war.

Der Novize programmierte Bewegungen (keine Punkte). Dazu ließ er den Roboter zunächst ebenfalls den nächsten Punkt anfahren und schrieb sich dessen Koordinaten in sein Notizbuch. Dann erstellte er im Panel die Struktur der nächsten Bewegung (Gerüst aus Klammern und Kommas) und definierte die Bewegungsart. Anschließend berechnete er aus seinen Notizen den Abstand zwischen neuem und vorangehendem Punkt und übertrug das Distanzmaß in die

Bewegungsstruktur. Falls es sich um eine Zirkularbewegung handelte, fügte auch er hier einen (allerdings berechneten) Wert für den Hilfspunkt ein. Ein weiterer entscheidender Unterschied ist, dass der Novize nach nahezu jeder programmierten Bewegung eine Kontrollfahrt durchführte, um die Genauigkeit zu überprüfen und bei Bedarf die Bahn zu korrigieren. Das wiederholte er nun, bis er am Ende der Außenbahn angelangt war.

Der nächste Schritt (jetzt in Block 2) lief bei beiden jeweils sehr ähnlich zum vorhergehenden ab. Um zum Startpunkt der Innenbahn zu gelangen, programmierte der Experte eine lineare Bewegung zu diesem ein (Übergang zwischen Außen- und Innenbahn). Ausgehend davon teachte er die weiteren Punkte genau wie zuvor. Nur beim letzten Punkt ging er anders vor. Er ließ sich die Koordinaten des Startpunktes der Außenbahn ausgeben, da dieser gleichzeitig auch den Endpunkt der Innenbahn darstellte. Die Werte gab er dann numerisch in das Inlineformular ein, sodass ein exakter Abschluss der Bahn gewährleistet wurde (zweiter Übergang). Dann fügte er wieder für jede Bewegungsart die Überschleifbefehle ein und startete eine Kontrollfahrt, bei der er Unstimmigkeiten identifizierte und korrigierte, bis er mit der Trajektorie zufrieden war. Der Novize fuhr die Innenbahn nicht wie der Experte zurück, sodass sie am Startpunkt der Außenbahn endete. Sondern er programmierte die Innenbahn parallel in dieselbe Richtung wie die Außenbahn. Dieses Vorgehen bot ihm den Vorteil, dass er nun den Code von der Außenbahn kopieren und für die Innenbahn nutzen konnte, da die Bahnen parallel verlaufen und daher bezüglich der Bewegungen prinzipiell das Gleiche geschieht. Nur die Werte musste er ersetzen, indem er wie zuvor die Punkte anfuhr und sich die Distanzen berechnete. Auch hier führte der Novize nach fast jeder neuen Bewegung eine Kontrollfahrt durch. Er arbeitete dabei nun parallel: nach einer Kontrollfahrt nahm er oftmals Korrekturen an der Außen- sowie der Innenbahn vor und erstellte gleich noch die neue Bewegung, um die Bahn fortzusetzen. Dies machte er solange, bis er am Ende angelangt war.

Zum Schluss setzten beide noch finale Einstellungen um, die nötig waren, damit der Roboter am Ende auch tatsächlich die Bahn in das Werkstück fräst. Der Experte musste dafür bei seiner Vorgehensweise die Höhe jedes einzelnen Punktes der Bahn innerhalb des gesamten Programms um den entsprechenden Wert (in diesem Fall 5 mm) heruntersetzen. Schlussendlich führte er noch eine Kontrollfahrt durch und korrigierte ggf. auftretende Fehler und wiederholt dies, bis er mit der gesamten Bahn zufrieden war und der Meinung, dass die Aufgabe hinreichend genau erfüllt war. Der Novize musste im letzten Schritt lediglich die Höhe des Startpunktes vom ursprünglich geteachten Wert um 5 mm herabsetzen. Anschließend wäre es sinnvoll gewesen noch Überschleifbefehle für alle Punkte einzufügen um die Bahn harmonischer (ohne Zwischenhaltpunkte) ablaufen zu lassen, was aber der Novize aus Zeitgründen in diesem Fall nicht vornahm. Nach Beendigung des Programmierens der Fräsbahnen wurden die Werkstücke gefräst.

4.2 Qualitativer Vergleich

Der qualitative Vergleich umfasst zum einen die Bewertung der Endergebnisse, sowie die zur Herstellung benötigte Bearbeitungszeit und die dabei aufgetretenen Fehler.

Die beiden erstellten Programme wurden von dem Roboter, mit dem die Trajektorien programmiert worden sind,

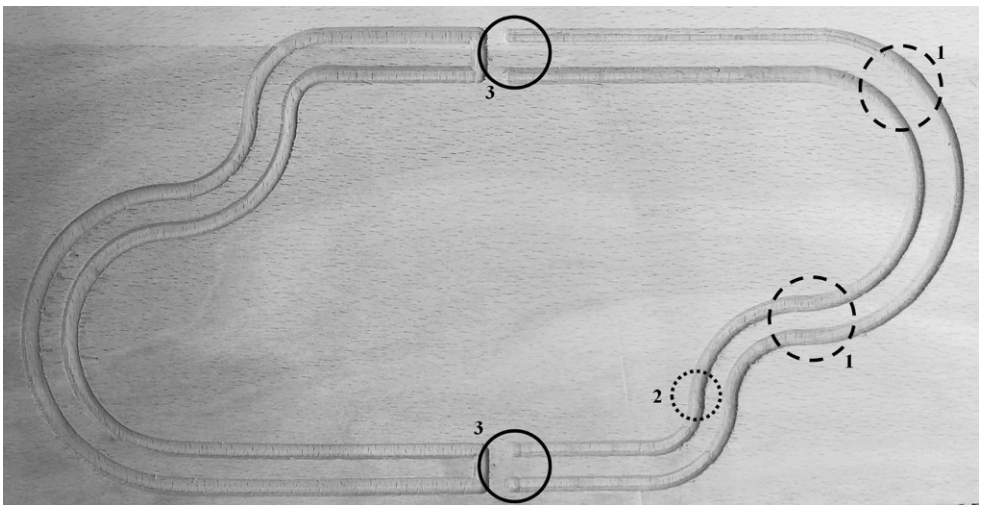

Abb. 4 Endergebnisse: Werkstück nachdem der Roboter die beiden Programme durchführte (*links* Experte, *rechts* Novize). Begutachtung durch Mitarbeiter des IWU, *1*: Schwankungen in der Breite; *2*: Absätze zwischen den einzelnen Bewegungen; *3*: fehlende Verbindungen zwischen Außen- und Innenbahn

Fig. 4 Final results: workpiece after the robot carried out the two programs (*left* expert, *right* novice). Assessment by IWU employees, *1*: fluctuations in width; *2*: irregularities between the individual movements; *3*: missing connections between outer and inner lanes

Tab. 2 Bearbeitungszeit der Aufgabe vom Experten und Novizen aufgelistet nach den einzelnen Schritten
Table 2 Processing time of the task by experts and novices listed according to the individual steps

Bearbeitungsschritt	Zeit Experte (in MM:SS)	Zeit Novize (in MM:SS)
(1) Vorbereitung: Vorüberlegung	00:25	01:50
(2) Vorbereitung: Einrichten	01:43	02:40
(3) Vorbereitung: Programm	03:36	01:55
Vorbereitung	*05:44*	*06:25*
(4) Programmieren: Außenbahn	08:16	24:25
(5) Programmieren: Innenbahn	09:50	20:10
(6) Programmieren: finale Einstellungen	09:40	03:30
Programmieren	*27:46*	*48:05*
Gesamt	**33:30**	**54:30**

ausgeführt. Als erstes Qualitätskriterium der Ergebnisse (Abb. 4) gaben der Gruppenleiter und ein weiterer unabhängiger Mitarbeiter des *Fraunhofer-IWU* eine Beurteilung zu den beiden Bahnen ab. Dabei stellte sich der Verlauf des Experten als besser im Vergleich zu der des Novizen heraus. In der Bahn des Novizen konnten mehrere größere Schwankungen in der Breite festgestellt werden (vgl. Abb. 4, Markierungen 1), wohingegen die Bahn des Experten in der Breite konstanter verlief, auch wenn hier ebenfalls kleinere Schwankungen sichtbar waren. Des Weiteren erkannten die Begutachter in der Bahn des Novizen teilweise kleinere Absätze zwischen den einzelnen Bewegungen (vgl. z. B. Abb. 4, Markierung 2), während die Bahn des Experten dahingehend flüssiger verlief. Diese Absätze könnten auf das Fehlen der Überschleifbefehle zurückzuführen sein, aufgrund dessen nach jeder Bewegung eine kurze Pause aufgetreten ist, während der Fräskopf auf der Stelle rotierte. Außerdem fällt auf, dass die Bahn des Novizen nicht vollständig ist; es fehlen die beiden Verbindungen zwischen Innen- und Außenbahn (vgl. Abb. 4, Markierungen 3). Ein weiterer Aspekt der Qualität bezieht sich auf die Ausführung des Programmierens, nicht auf das Endergebnis: die Bearbeitungszeit unterscheidet sich deutlich zwischen dem Experten und dem Novizen (siehe Tab. 2). Es ist zu erkennen, dass beide zur Vorbereitung in etwa dieselbe Zeit benötigten. Der Novize brauchte für die Vorüberlegung ein wenig länger, da er sich hier gegenüber dem Experten noch Informationen aus dem Handbuch holen musste. Das Einrichten dauerte bei ihm ebenfalls ein bisschen länger eventuell aus Gründen der mangelnden Routine. Beim Vorbereiten des Programms war es umgekehrt, da der Experte hier noch einen kleinen Testdurchgang durchführte. Der erste Block (Vorbereitung) sah insgesamt bei beiden Versuchspersonen sehr ähnlich aus. Wesentliche Unterschiede traten erst beim eigentlichen Programmieren im zweiten Block auf. Sowohl für die

Tab. 3 Anzahl Fehler nach Versuchsperson und Fehlertyp
Table 3 Number of errors according to participant and error type

Fehlertyp	Experte	Novize
Patzer: Zeitmissmanagement	0	1
Patzer: Vertauschung	0	1
Patzer: Unterlassung	1	2
Patzer: Störung	0	15
Schnitzer: Verlust des aktuellen Stands	0	3
Regelbasierter Fehler	3	5
Gesamt	*4*	*27*

Außen- als auch für die Innenbahn brauchte der Novize wesentlich länger als der Experte, was durch den zeitlichen Vorteil bei den finalen Einstellungen (Punkt 6 in Tab. 2/Abb. 2) nicht aufgeholt wurde. Der Hauptgrund für diese zeitliche Differenz sind die vielen Kontrollfahrten, die der Novize durchführte. Wie oben bereits beschrieben ging der Novize nach fast jeder neuen Bewegung in eine solche Überprüfung und nahm daraufhin auch entscheidend mehr Korrekturen bzw. Anpassungen vor. In Zahlen: der Novize absolvierte 26 Kontrollfahrten, welche insgesamt ca. 10 min seiner Bearbeitungszeit in Anspruch nahmen. Der Experte hingegen kam mit acht Kontrollfahrten und einer Zeit von ca. 4 min aus. Die genaue Bearbeitungszeit beim Durchführen der Korrekturen und Anpassungen lässt sich nur schwer von anderen Bearbeitungsschritten trennen. Jedoch ließ sich feststellen, dass der Novize 22-mal Veränderungen vornahm, der Experte im Gegensatz dazu nur viermal.

Man kann jede dieser Anpassungen auf einen vorangegangenen Fehler zurückführen, welche nach Reason (1990) wie oben beschrieben eingeteilt werden können. Die Anzahl nach Fehlertypen für Experte und Novize können Tab. 3 entnommen werden. Insgesamt beging der Novize 27 Fehler, der Experte hingegen nur vier. Die Patzer können in unterschiedliche untergeordnete Fehlertypen eingeordnet werden. Der Novize zeigte zeitliches Missmanagement: er benötigte 54 statt der vorgegebenen 30 min. Der Experte lag mit etwa 33 min nur leicht drüber und ihm wird kein zeitliches Missmanagement zugeschrieben. Des Weiteren unterlief dem Novizen eine Vertauschung (falschen Wert ins Notizbuch übertragen, vgl. Punkt 4.2.2 bzw. 5.3.2 in Abb. 3) und zwei Unterlassungen (Vorzeichen vergessen und Verbindungen zwischen Außen- und Innenbahn). Der Experte hingegen hatte nur eine Auslassung (einen Punkt beim tiefer setzen vergessen, vgl. Punkt 6.2.3 in Abb. 2). Der größte Unterschied ist bei den Störungen zu sehen, welche beim Experte gar nicht vorkommen und beim Novizen 15-mal (z. B. Bewegungsbahn zu ungenau gesetzt, sodass Korrektur notwendig war, vgl. Punkt 4.2 bzw. 5.3 in Abb. 3). Schnitzer in Form von Verlust des aktuellen Standes zeigten sich beim Novizen dreimal (Austritt aus der Roboterzelle, um im Manual erneut nach Informationen zu suchen, vgl.

Punkt 1.2 in Abb. 3). Regelbasierte Fehler traten bei beiden Versuchspersonen auf, beim Experten dreimal (z. B. Überschleifbefehl falsch umgesetzt, vgl. Punkt 4.3 bzw. 5.4 in Abb. 2) und beim Novizen fünfmal (z. B. Hilfspunkt in einer Kurve falsch gesetzt, vgl. Punkt 4.2.6 in Abb. 3).

Alles in allem betrachtet arbeitete der Experte (erwartungsgemäß) konsistenter. Durch konsequente Fehlervermeidung war es ihm möglich, eine höhere Qualität im Endergebnis der Bahn mit geringerem Zeitaufwand zu erreichen.

4.3 Kognitive Prozesse beim Teach-in

Es konnten alle unterbewussten kognitiven Prozesse sowie folgende metakognitive Prozesse in Anlehnung an Renumol et al. (2010) anhand der Auswertung der Videos und Interviewtranskripte identifiziert werden: Aufmerksamkeitsmechanismen, Wissensrepräsentation und visuelle Suche. Bei den höheren kognitiven Prozessen zeigten sich zehn von 16: Erkennen, Verbildlichen, Verständnis, Lernen, Entscheidungsfindung, Problemlösen, Analyse, Synthese, Kreieren und Planen. Die Hälfte der zusätzlichen kognitiven Prozesse wurde auch in dieser Untersuchung gefunden: Hypothesenbildung, Iterieren, Überwachen und Erinnern.

Tab. 4 Beschreibung der *Areas of Interest* (AOIs)
Table 4 Description of *Areas of Interest* (AOIs)

Bezeichnung	Beinhaltet
Roboter	Roboterfuß, Gelenke des Roboterarms
Werkzeug	Fräs„spitze, Fassung des Fräsers im Roboter
Werkbank	Werkstück, Aufgabenblatt (Schablone), Tisch auf dem gearbeitet wurde
Panel	Gesamtes Panel mit 6D-Maus
Manual (nur Novize)	Computer außerhalb der Roboterzelle mit Bedienanleitung in *pdf*-Format, Notizbuch
Sonstige	Alle Fixationen, die keinem der anderen Bereiche zugeordnet werden können

4.4 Augenbewegungscharakteristik

Einige kognitive Prozesse können anhand von bestimmten Verhaltensparametern beschrieben und untersucht werden. Im folgenden Abschnitt wird auf Daten aus der Augenbewegungsmessung eingegangen, welche Aufschluss bspw. über visuelle Suche oder Aufmerksamkeitsmechanismen geben könnten (vgl. z. B. Armstrong und Olatunji 2012; Bucher und Schumacher 2006; Mele und Federici 2012; Najemnik und Geisler 2005). Vorab mussten dazu *Areas of Interest* (AOIs) definiert werden: Bereiche, die bei der Bearbeitung der Programmieraufgabe von besonderer Bedeutung sind und häufiger angeguckt wurden (siehe Tab. 4).

In Tab. 5 sind die Schlüsselindikatoren der Eye-Tracking Daten aufgelistet (ausgewertet mit der Software *BeGaze*, siehe Gaze Intelligence 2020). Sowohl Experte als auch Novize schauen mit ähnlichem prozentualen Anteil am längsten auf das AOI „Panel“ (46,2 % vs. 45,8 %) und am kürzesten auf das AOI „Roboter“. Für die anderen vier AOI sind Abweichungen voneinander vorzufinden. Die größte Differenz von 11,7 % liegt beim „Werkzeug“ vor. Einen weiteren großen Unterschied sieht man beim „Manual“, da der Experte weder in die Bedienanleitung noch auf ein Notizbuch geschaut hat, während der Novize dies relativ häufig nutzte. Die Abweichungen in den AOIs „Werkbank“ und „Sonstige“ fallen mit 2,8 und 3,9 % etwas geringer aus.

Ein weiterer interessanter Vergleichspunkt ist die durchschnittliche Dauer der Fixationen pro AOI. Beide Versuchspersonen zeigen mit etwa 610 ms die größte Ausprägung auf dem AOI „Werkzeug“. Größere Differenzen in der Dauer der Fixationen zeigen sich bei den AOIs „Roboter“ und „Werkbank“ (ca. 130 ms bzw. 100 ms). Kleiner fallen die Unterschiede für „Panel“ und „Sonstige“ aus (jeweils ca. 30 ms).

Tab. 5 Schlüsselindikatoren der Eye-Tracking Daten für Experte und Novize nach AOIs
Table 5 Key indicators of eye tracking data for expert and novice according to AOIs

		Roboter	Panel	Werkzeug	Werkbank	Sonstige	Manual
Experte	Verweildauer	4,3 s (0,2 %)	950,0 s (46,2 %)	675,6 s (32,8 %)	180,3 s (8,8 %)	113,6 s (5,5 %)	–
	Durchschnittliche Dauer Fixation	169,1 ms	278,5 ms	614,6 ms	316,5 ms	239,4 ms	–
	Anzahl Fixation	21	2966	1050	510	385	–
	Rückkehr	13	118	197	134	48	–
	Anzahl/Rückkehrer	1,62	25,14	5,32	3,81	8,02	–
Novize	Verweildauer	11,6 s (0,4 %)	1400,1 s (45,8 %)	644,5 s (21,1 %)	355,8 s (11,6 %)	49,7 s (1,6 %)	321,7 s (10,5 %)
	Durchschnittliche Dauer Fixation	294,9 ms	306,7 ms	606,1 ms	412,2 ms	269,5 ms	311,4 ms
	Anzahl Fixation	35	4041	1021	818	150	919
	Rückkehr	9	233	241	255	32	78
	Anzahl/Rückkehrer	3,89	17,34	4,27	3,21	4,69	11,78

5 Diskussion

Bei der Bearbeitung einer Aufgabe zum Programmieren eines Industrieroboters durch einen Experten und einen Novizen wurden mit Hilfe von retrospektiven *Think-Aloud* Interviews und Eye-Tracking formalisierte Prozessbeschreibungen des Einteachens erstellt (vgl. Abb. 2 und 3). Die beiden unterschiedlichen Herangehensweisen von Experte vs. Novize wiesen einige Gemeinsamkeiten auf. Beispielsweise lassen sich beide Vorgehensweisen grob in die gleichen Phasen einteilen: Vorbereitung und Programmieren. Auch in den Teiltätigkeiten gibt es Ähnlichkeiten, z. B. Vorüberlegung, Einrichten der Umgebung, Vorbereiten des Programms. Diese Unterteilung scheint allgemein sinnvoll zu sein. Hingegen zeigen sich beim Teilschritt „Programmieren" Unterschiede in Abhängigkeit davon, was in der konkreten Aufgabe gefordert ist (z. B. Programmieren von (Relativ-)Bewegungen über manuelle Eingabe ins Panel vs. Teachen von Punkten über Inlineformulare).

Der Experte arbeitete erwartungsgemäß mit höherer Qualität und Effizienz. Die von ihm erstellte Bahn war insgesamt sauberer, geradliniger und konsistenter als die des Novizen, welcher auch einen Teil der Bahn nicht fertigstellte. Dabei benötigte der Experte deutlich weniger Zeit zum Absolvieren der Aufgabe und beging währenddessen weniger Fehler. Der Novize baute in sein Vorgehen sehr viele Kontrollen ein und besserte zwischendurch immer wieder nach, was sich letztendlich in der deutlich höheren Bearbeitungszeit niederschlug.

Des Weiteren wurden kognitive Prozesse identifiziert, welche beim Teach-in des Industrieroboters von Bedeutung waren. Hierzu zählen vor allem visuelle Informationssuche und Aufmerksamkeitsprozesse, die sich auch in den Ergebnissen der Eye-Tracking Daten näher beleuchten lassen. Es gibt z. B. Unterschiede in den Parametern Verweildauer, durchschnittliche Fixationsdauer und Anzahl der Fixationen und Rückkehrer. Diese können entweder zwischen den AOIs auftreten oder aber zwischen den Probanden. Aus den vorliegenden Eye-Tracking Daten können erste hypothesenbildende Aussagen abgeleitet werden: Zum einen ist zu erkennen, dass der Experte eine geringere durchschnittliche Fixationsdauer über alle AOIs hinweg im Gegensatz zum Novizen aufweist. Dies liegt möglicherweise an der schnelleren Verarbeitung von domänenspezifischen Informationen durch entsprechend ausgebildete Wissensstrukturen (z. B. Chi et al. 1982; Gobet 1998; Gobet und Simon 1996), weswegen Experten weniger Zeit benötigen sich diese Informationen aus der Umwelt zu verschaffen und sie zu deuten. Größere Differenzen in der durchschnittlichen Fixationsdauer gibt es bei den AOIs „Werkbank" mit knapp 100 ms und „Panel" mit 30 ms. Diese beiden AOIs könnten demnach besonders interessant sein, was domänenspezifische Informationen anbelangt. Das Panel ist das Hauptmedium, mit dem der Programmierer in Interaktion mit dem Roboter tritt. Gerade hier werden also viele programmierrelevante Informationen dargeboten. Auf der Werkbank hingegen befindet sich das Werkstück, welches vor allem aufgabenspezifische Informationen bereithält, je nachdem was gemacht werden soll (Schweißen, Fräsen, Lasern etc.). Hier ist wahrscheinlich auch die Erfahrung des Programmierers speziell mit der Art der Anwendung wesentlich.

Weiterhin sind die Werte der Variable „Anzahl pro Rückkehrer" über alle AOIs beim Experten höher ausgeprägt als beim Novizen, d. h. er wechselte weniger häufig zwischen den einzelnen AOIs hin und her. Auch das könnte für eine schnellere Informationsverarbeitung sprechen bzw. für eine höhere Arbeitsgedächtnisleistung in der entsprechenden Domäne bspw. aufgrund größerer *Chunks* (z. B. Gobet und Clarkson 2004; Ye und Salvendy 1994), aufgrund derer er die Informationen länger aufrechterhalten konnte und nicht so oft wie der Novize sich diese wieder aus der Umgebung beschaffen musste. Für diesen Parameter ist der größte Unterschied im Panel zu finden, welches wie oben beschrieben das wichtigste Instrument des Programmierers beim Teach-in ist.

Die große Abweichung in der Verweildauer von über 10 % beim AOI „Werkzeug" könnten damit zusammenhängen, dass der Experte bei Kontrollfahrten einen anderen Fokus als der Novize legte, welcher dabei mehr zwischen Werkzeug und Werkbank wechselte. Oder aber der Unterschied ist darauf zurückzuführen, dass er bei seiner Vorgehensweise (die Punkte anfahren und direkt einspeichern) mehr auf die Werkzeugspitze als Orientierung angewiesen war als der Novize. Ob die verschiedenen Vorgehensweisen die Hauptursache für die genannten Unterschiede sind, muss in weiteren Studien überprüft werden. Eventuell gibt es auch noch weitere Arten der Aufgabenbearbeitung in Abhängigkeit von verschiedenen Einflussfaktoren (Expertisegrad, Qualifikationshintergrund usw.).

Der Experte provozierte in seinem Vorgehen lediglich vier Fehler Beim Novizen hingegen konnten 27 Fehler identifiziert werden. Dieser große Unterschied in der Anzahl der Fehler, die zunächst als solche identifiziert und anschließend korrigiert werden mussten, trug ebenfalls zu der deutlich höheren Bearbeitungszeit des Novizen bei. Es sind v. a. Patzer und Schnitzer die ihm häufiger unterlaufen. Auf der fähigkeitsbasierten Ebene im Bereich der Aufmerksamkeit und des Gedächtnisses könnte dem Novizen dahingehend mehr Unterstützung bzw. Assistenz geliefert werden, um sowohl Effizienz als auch Qualität in der Bearbeitung zu steigern. Einer weiteren Alternativerklärung könnte in zukünftigen Studien nachgegangen werden: inwieweit spielt das Selbstvertrauen des Programmierers oder Vertrautheit mit der spezifischen Aufgabe eine Rolle hinsichtlich des Begehens von Fehlern?

Die dargestellte methodische Vorgehensweise zur Formalisierung von Wissen beim Teach-in auf Basis von retrospektiven *Think-Aloud* Interviews mit Eye-Tracking hat sich bewährt. Die Ergebnisse zeigen eine erste solide Systematisierung, die eine Grundlage für weitere Studien bildet. Limitiert ist die Studie aufgrund der geringen Probandenanzahl (ein Proband pro Gruppe Experte vs. Novize). Unterschiede in den Parametern zwischen den beiden könnten auch auf andere Ursachen zurückführbar sein, z. B. auf interindividuelle Unterschiede, die nicht im Zusammenhang mit der Erfahrung oder dem Wissen stehen. Auch wenn die qualitativen und deskriptiven Ergebnisse noch nicht generalisierbar sind, so sind sie dennoch hilfreich für die Bildung von Hypothesen, die mit einer größeren Stichprobe weiter untersucht werden können.

Die formalisierte Beschreibung bezieht sich zunächst nur auf die spezifische Aufgabe, die die beiden Versuchspersonen absolvierten. Es ist zu prüfen, inwieweit auch andere Programmieraufgaben einen ähnlichen schematischen Ablauf zeigen. Für diese spezifische Aufgabe konnten bereits einige kognitive Prozesse aus der Liste von Renumol et al. (2010) identifiziert werden, die bei der Bearbeitung eine Rolle spielen. Ob diese und eventuell weitere kognitive Prozesse auch relevant für andere Aufgaben im Bereich des Teach-ins von Industrierobotern sind, könnte in zukünftigen Studien untersucht werden.

Im Zusammenhang der Betrachtung weiterer Aufgaben kann die Aufgabenschwierigkeit eine Rolle spielen. Die Probanden schätzten die Aufgabe als mittelschwer ein, was auch mit Randbedingungen wie gefühltem Zeitdruck bei der Bearbeitung zusammenhing. Neben der Schwierigkeit können weiterhin andere Eigenschaften das Vorgehen zur Programmieraufgabe beeinflussen wie bspw. das vom Roboter auszuführende Fertigungserfahren (Handhaben, Fügen, Umformen, Fräsen, Lackieren, Laserschneiden etc.) und das damit zusammenhängende Tool oder auch der Umfang des zu bearbeitenden Projekts und die daraus resultierenden zeitlichen Maßstäbe.

Die vorliegende Arbeit ist ein erster Schritt in Richtung Guidelines, wie man Anfänger im Bereich des Programmierens und des Teach-ins von Industrierobotern den Einstieg erleichtern kann bzw., wie man diese schneller den Status eines Experten erreichen könnten. Auf Grundlage der Erkenntnisse über die unterschiedlichen Vorgehensweisen von Novizen und Experten und durch Einbezug der in diesem Feld relevanten kognitiven Prozesse kann man die Mensch-Roboter-Interaktion verbessern. Bspw. könnte man die technische Bedienung des Roboters (*Touch-Panel*, GUI, etc.) auf das entsprechende Level des jeweiligen Nutzers anpassen oder auch ein spezifisches Training für verschiedene Level entwickeln. Auf diese Weise könnte man die Novizen dahingehend unterstützen, dass sie schneller eine effizientere Vorgehensweise entwickeln und somit ihren Expertisegrad erhöhen. Um diesem Ziel näher zu kommen, ist allerdings noch weiterer Forschungsbedarf vorhanden. Stufen des Fähigkeitserwerbs im Bereich Programmieren bzw. Teach-in sollten identifiziert werden. Darauf aufbauend kann der Frage nachgegangen werden, wo man ansetzen könnte, um Probanden auf die nächste Stufe zu bringen bzw. welche kognitiven Ressourcen besonders geschult werden könnten. Weiterhin sollte geklärt werden, welche konkreten Gedächtnisressourcen in diesem Feld benötigt werden. So könnten die für das Programmieren wichtigsten Informationen aus der Umwelt herausgefiltert werden. Auch die Berücksichtigung des Einflusses unterschiedlicher Programmiermethoden (Heimann und Guhl 2020) ist in diesem Zusammenhang wichtig. Ebenso sollte erforscht werden, inwiefern neue Arten der Programmierung, z. B. „programming by demonstration“ (z. B. Pinto et al. 2020) zu realen Vereinfachungen im Teach-in Prozess führen und z. B. die kognitive Belastung oder den mentalen Workload des Operators optimieren helfen.

Danksagung Wir danken Herrn Victor Wiedemann und Herrn Sebastian Mach für die Mitwirkung bei der Datenerhebung.

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

Anderson JR (2007) Expertise. In: Funke J (Hrsg) Kognitive Psychologie. Spektrum Akademischer Verlag, Heidelberg, S 331–366

Armstrong T, Olatunji BO (2012) Eye tracking of attention in the affective disorders: a meta-analytic review and synthesis. Clin Psychol Rev 32(8):704–723

Bloom BS (Hrsg) (1956) Cognitive domain. Taxonomy of educational objectives, Bd. 1. McKay, New York

Bocklisch F, Drehmann R, Lampke T (2020) Kognitionsbasierte Mensch-Technik Interaktion in Cyber-Physischen Systemen am Applikationsbeispiel „Thermisches Spritzen“. TU Chemnitz, Chemnitz

Bocklisch F, Paczkowski G, Zimmermann S, Lampke T (under review) Integration human cognition in cyber-physical systems: a multidimensional fuzzy pattern model with application to thermal spraying. Journal of Manufacturing Systems (submitted)

Botthof A, Hartmann EA (Hrsg) (2015) Zukunft der Arbeit in Industrie 4.0. Springer Vieweg, Wiesbaden
Bucher HJ, Schumacher P (2006) The relevance of attention for selecting news content. An eye-tracking study on attention patterns in the reception of print and online media. Communications 31(3):347–368
Charmaz K, Belgrave L (2012) Qualitative interviewing and grounded theory analysis. In: Gubrium JF, Holstein JA, Marvasti AB, McKinney KD (Hrsg) The SAGE handbook of interview research: the complexity of the craft, Bd. 2. SAGE, Thousand Oaks, S 347–365
Chase WG, Simon HA (1973) Perception in chess. Cogn Psychol 4(1):55–81
Chi MT, Glaser R, Rees E (1981) Expertise in problem solving. Pittsburgh University Learning Research and Development Center, Pittsburgh
Davies SP (1993) Models and theories of programming strategy. Int J Man Mach Stud 39(2):237–267
DIN (2020) Was ist Industrie 4.0?. https://www.din.de/de/forschung-und-innovation/themen/industrie4-0/was-ist-industrie-4-0--73174. Zugegriffen: 06. Mai 2020
Feigenspan J, Kästner C, Liebig J, Apel S, Hanenberg S (2012) Measuring programming experience. 20th IEEE International Conference on Program Comprehension (ICPC), IEEE, Passau, S 73–82
Gaze Intelligence (2020) SMI Software. https://gazeintelligence.com/smi-software-download. Zugegriffen: 23. Aug. 2020
Gobet F (1998) Expert memory: a comparison of four theories. Cognition 66(2):115–152
Gobet F, Clarkson G (2004) Chunks in expert memory: Evidence for the magical number four... or is it two? Memory 12(6):732–747
Gobet F, Simon HA (1996) Templates in chess memory: a mechanism for recalling several boards. Cogn Psychol 31(1):1–40
Götz DA (2020) Abbildung von menschlichem Expertenwissen und kognitiven Prozessen beim Teach-in von Industrierobotern. Unpublished Master Thesis, Chemnitz University of Technology
Heimann O, Guhl J (2020) Industrial robot programming methods: a scoping review. 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, Wien, S 696–703
Hermann M, Pentek T, Otto B (2016) Design principles for industrie 4.0 scenarios. 49th Hawaii international conference on system sciences (HICSS), IEEE, Manoa, S 3928–3937
Kagermann H, Lukas WD, Wahlster W (2011) Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. VDI Nachr 13:1
Kagermann H, Wahlster W, Helbig J (2013) Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Frankfurt/Main. Plattform Industrie 4.0
Kasto N (2012) The development of knowledge in novice programmers. In: Proceedings of the ninth annual international conference on International computing education research. Association for Computing Machinery, San Diego, S 159–160
Krems J (1994) Wissensbasierte Urteilsbildung: diagnostisches Problemlösen durch Experten und Expertensysteme. Huber, Bern
KUKA (2020a) KUKA. SystemSoftware. https://www.kuka.com/de-de/produkte-leistungen/robotersysteme/software/systemsoftware/kuka_systemsoftware. Zugegriffen: 20. Juli 2020
KUKA (2020b) KR 150 R2700 extra. https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/0000182737_en.pdf. Zugegriffen: 23. Aug. 2020
Lee EA (2008) Cyber physical systems: design challenges. 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), IEEE, Orlando. IEEE, USA, S 363–369
Lee J, Bagheri B, Kao HA (2015) A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manuf Lett 3:18–23
Madni AM (2018) Transdisciplinary systems engineering: exploiting convergence in a hyper-connected world. Springer, Cham
Madni AM (2019) Adaptive cyber-physical-human systems: a 21st century perspective. 2nd international conference on Intelligent Human Systems Integration, IHSI, San Diego, S 1535–1553
Mele ML, Federici S (2012) Gaze and eye-tracking solutions for psychological research. Cogn Process 13(1):261–265
Miller GA (1956) The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychol Rev 63(2):81
Najemnik J, Geisler WS (2005) Optimal eye movement strategies in visual search. Nature 434(7031):387–391
Nelles J, Kuz S, Mertens A, Schlick CM (2016) Human-centered design of assistance systems for production planning and control: The role of the human in Industry 4.0. 2016 IEEE International Conference on Industrial Technology (ICIT), IEEE, Taipei. IEEE, S 2099–2104
Parnin C, Siegmund J, Peitek N (2017) On the nature of programmer expertise. Psychology of programming interest group, S 16
Peitek N, Siegmund J, Apel S (2020) What drives the reading order of programmers? An eye tracking study. Proceedings of the 28th International Conference on Program Comprehension, Seoul, S 342–353
Pinto VH, Amorim A, Rocha L, Moreira AP (2020) Enhanced performance real-time industrial robot programming by demonstration using stereoscopic vision and an IMU sensor. International Conference on Autonomous Robot Systems and Competitions, Ponta Delgada, S 108–113
Plattform-I40 (2019) Working Paper Technologieszenario „Künstliche Intelligenz in der Industrie 4.0“. https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/KI-industrie-40.pdf?__blob=publicationFile&v=10. Zugegriffen: 28. Okt. 2020
Plattform-I40 (2020) Was ist Industrie 4.0? https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html. Zugegriffen: 6. Mai 2020
Posner MI (1988) Introduction: What is it to be an expert? In: Chi MTH, Glaser R, Farr MJ (Hrsg) The nature of expertise. Erlbaum, Hillsdale, S 29–36
Reason JT (1990) Human error. Cambridge University Press, NY, S 1–96
Renumol VG, Janakiram D, Jayaprakash S (2010) Identification of cognitive processes of effective and ineffective students during computer programming. Acm Trans Comput Educ 10(3):1–21
Ruckpaul A, Fürstenhöfer T, Matthiesen S (2015) Combination of eye tracking and think-aloud methods in engineering design research. In: Sixth International Conference on Design Computing and Cognition, DCC'14. Springer, London, S 81–97
Van Gog T, Paas F, Van Merriënboer JJ, Witte P (2005) Uncovering the problem-solving process: Cued retrospective reporting versus concurrent and retrospective reporting. J Exp Psychol Appl 11(4):237–244
Van Den Haak M, De Jong M, Schellens JP (2003) Retrospective vs. concurrent think-aloud protocols: testing the usability of an online library catalogue. Behav Inf Technol 22(5):339–351
Vessey I (1985) Expertise in debugging computer programs: a process analysis. Int J Man Mach Stud 23(5):459–494
Wang F-Y (2010) The emergence of intelligent enterprises: from CPS to CPSS. IEEE Intell Syst 25(4):85–88
Wang Y, Patel S, Patel D, Wang Y (2003) A layered reference model of the brain. In: Proceedings of the Second IEEE International Conference on Cognitive Informatics (ICCI), IEEE London, S 7–17
Whalley J, Kasto N (2014) A qualitative think-aloud study of novice programmers' code writing strategies. In: Proceedings of the 2014 conference on Innovation & technology in computer science education Uppsala, S 279–284
Wickens CD, Gordon SE, Liu Y (1998) Cognition. In: McGeehon P (Hrsg) An introduction to human factors engineering. Addison-Wesley, New York, S 145–182
Ye N, Salvendy G (1994) Quantitative and qualitative differences between experts and novices in chunking computer software knowledge. Int J Hum Comput Interact 6(1):105–118