Johner Institut

Codegenerierung bei Medizinprodukten

Mit Matthias van der Staay, Prof. Dr. Christian Johner

Transkript

00:00:05 Sprecher 1

Medical Device Insights, ein Podcast des Ione Instituts für Medizinproduktehersteller, Behörden und benannte Stellen.

00:00:18 Sprecher 2

Bei der Entwicklung von Medizinprodukten spielt Software eine immer größere Rolle, wie Sie wissen.

00:00:23 Sprecher 2

Also nicht nur, dass Software Teil des Medizinprodukts ist, oft ist ja

00:00:27 Sprecher 2

die Software selbst des Medizinprodukts.

00:00:30 Sprecher 2

Gerade angesichts der Tatsache, dass es auch immer schwerer wird, Expertinnen und Experten zu finden, die Software, gute Software schreiben können, stellt sich die Frage, ob wir mit Softwaregenerierung nicht einen möglichen Ausweg aus diesem Dilemma finden.

00:00:46 Sprecher 2

Und deswegen wollen wir uns in dem heutigen Podcast anschauen, was bedeutet Code-Generierung, wann ergibt es einen Sinn, wie gibt man da, geht man dabei vor

00:00:57 Sprecher 2

Und auf was sollte man das speziell bei der Entwicklung von Medizinprodukten achten?

00:01:01 Sprecher 2

Und dazu hab ich einen Experten eingeladen, nämlich der Matthias van der Stey von der Firma I.M.T.

00:01:07 Sprecher 2

Das ist ein Schweizer Unternehmen, auch Entwicklungsdienstleister, ja, mit dem wir seit Jahren zusammenarbeiten.

00:01:14 Sprecher 2

Ja, Herr van der Stey, wenn Sie sich kurz vorstellen, vielleicht auch von der Ausbildung und was Sie im

Kontext Code-Generierung machen, dann wird es nun zum guten Einstieg in das Thema geben.

00:01:25 Sprecher 3

Ja, mein Name ist Matthias van der Stey und ich arbeite nun seit über 10 Jahren bei der Firma I.M.T.

00:01:32 Sprecher 3

Gestartet habe ich dort im Bereich von Signalverarbeitung und Regelungstechnik und konnte in diversen Projekten mitarbeiten.

00:01:42 Sprecher 3

Auch Projekte in der Medizintechnik, also beispielsweise an einem Beatmungsgerät, habe ich mitgearbeitet.

00:01:51 Sprecher 3

Ein wenig später durfte ich die Teamleitung übernehmen für das Signalverarbeitungsteam und wir beschäftigen uns hauptsächlich mit Regelungstechnik und Signalverarbeitung in den Bereichen von Reglerdesign und Signalverarbeitung, also die ganzen Filterauslegungen und adaptive Verfahren.

00:02:13 Sprecher 3

Aber auch im Bereich von Bildverarbeitung und Machine Learning lösen wir unsere Probleme.

00:02:20 Sprecher 2

Und dabei, wenn ich es richtig verstehen stehe, wenden Sie dann auch eben die Verfahren der Code-Generierung auch direkt mit an, oder?

00:02:28 Sprecher 3

Genau, das ist so, vor allem im Kontext der Regelungstechnik, jetzt weniger im Bereich von Machine Learning.

00:02:34 Sprecher 3

Aber im Bereich von Regelungstechnik ist es ja sehr viele machen das, sehr viele Industriezweige verwenden da die Code-Generierung, weil es eben sehr gut in den ganzen Entwicklungsprozess passt.

00:02:47 Sprecher 2

Ja, schauen wir uns vielleicht mal an, was Code-Generierung überhaupt bedeutet.

00:02:51 Sprecher 2

Also, der Name legt natürlich nach.

00:02:52 Sprecher 2

Am Ende hat man irgendwie Code, der von der Maschine geschrieben worden ist.

00:02:56 Sprecher 2

Aber woraus oder was ist sozusagen der Input, mit dem wahrscheinlich ein anderes Stück Software dann den Quellcode schreibt?

00:03:04 Sprecher 2

Also, was ist sozusagen der Startpunkt von all dem?

00:03:07 Sprecher 3

Ja, also Code-Generierung ist grundsätzlich sehr vielfältig, wenn man es formell ausdrücken würde.

00:03:16 Sprecher 3

geht es einfach da drum, aus etwas Abstraktem, etwas weniger Abstraktem zu generieren.

00:03:23 Sprecher 3

Also das kann auch ein Compiler sein, weil dieser generiert auch aus C.

00:03:28 Sprecher 3

Code beispielsweise Maschinencode.

00:03:32 Sprecher 3

Aber man unter dem Begriff Code-Generierung versteht man auch häufig, dass man aus Modellen Code generiert.

00:03:40 Sprecher 3

Das heißt, aus einem abstrakten Modell wird dann beispielsweise Programm

00:03:46 Sprecher 3

Code erzeugt.

00:03:47 Sprecher 2

Was für Modelle haben Sie da?

00:03:49 Sprecher 2

Also, liegt es irgendwie in einem Tool vor, wie Enterprise Architect, oder wie erstellen Sie diese Modelle?

00:03:54 Sprecher 3

Ja, da gibt es verschiedene Tools, die wir da verwenden.

00:03:58 Sprecher 3

Also einerseits verwenden wir sehr stark die die Simulink Tools von Mathwerks zusammen mit Matlab, aber wir verwenden auch unsere eigenen Tools, die wir selbst geschrieben haben, zwecks der Code-Generierung.

00:04:12 Sprecher 2

Wenn wir über diese eigenen Tools sprechen,

00:04:15 Sprecher 2

Was nutzen Sie da als Input?

00:04:16 Sprecher 2

Ist es eine visuelle Beschreibung von Modellen oder haben Sie da eine eigene Sprache für entwickelt, um daraus letztlich den Code zu erzeugen?

00:04:26 Sprecher 3

Ja, das ist grundsätzlich beides.

00:04:28 Sprecher 3

Also, man hat die Wahl, ob man visuell etwas beschreiben möchte, aber man hat auch die Wahl, dass in einem Ort Konfigurationsdatei zu hinterlegen, was generiert werden soll und

00:04:44 Sprecher 3

Insofern ist man da keinem gewissen Prozess unterlegen.

00:04:50 Sprecher 2

Was Sie ja schon gesagt haben, ist, dass Sie die Code-Generierung vor allem im Kontext von Signalverarbeitung mit einsetzen.

00:04:59 Sprecher 2

Jetzt haben Sie uns auf der anderen Seite geschildert, Sie haben zwar auch kommerzielle Tools im Einsatz, aber auch welche geschrieben.

00:05:05 Sprecher 2

Das heißt, jetzt haben wir so eine Balance.

00:05:08 Sprecher 2

Auf der einen Seite muss Arbeit aufgewendet werden, um Code-Generatoren zu schreiben.

00:05:13 Sprecher 2

Auf der anderen Seite müsste alternativ Arbeit aufgewendet, um den Source Code zu schreiben.

00:05:18 Sprecher 2

Jetzt stellt sich die Frage, wann rechnet sich denn so ‚ne Investition?

00:05:22 Sprecher 2

In welchen Bereichen würden Sie besonders empfehlen, dass man Code generiert?

00:05:27 Sprecher 3

Ja, das ist grundsätzlich eine sehr gute Frage oder eine sehr vielseitige Frage, wann sich es rechnet und wann es Sinn macht.

00:05:35 Sprecher 3

Also für I.M.T.

00:05:37 Sprecher 3

war, ich meine, das Schreiben eines Code-Generators

00:05:42 Sprecher 3

Nur für ein einziges Produkt, das wird sich nie rechnen, weil da ist der Aufwand viel zu hoch.

00:05:48 Sprecher 3

Aber da IMT ja im Projektgeschäft tätig ist und IMT sehr viele Projekte umsetzen darf und dabei immer wieder die gleichen Arbeiten anstehen, macht es Sinn, diese Arbeiten zu automatisieren, zu vereinfachen und schlussendlich uns da effizienter zu machen.

00:06:07 Sprecher 2

Angenommen, wenn man so einen Code-Generator hat, das heißt, man hat mal diese Anfangsinvestition, diese allgemeine, getätigt, dann steht ja trotzdem noch die Arbeit an, mit Hilfe dieses Code-Generators erstmal die Modelle zu erzeugen.

00:06:21 Sprecher 2

Ja, das heißt, dann haben wir jetzt eine andere Balance zu finden, nämlich zwischen der Modellierung im Modell und auf der einen Seite und dem Schreiben des Codes.

00:06:30 Sprecher 2

Gibt es Bereiche, wo Sie sagen würden, dass die Modellierung einfach viel schneller geht als dieses Schreiben des Quellcodes von Hand?

00:06:38 Sprecher 3

Ja, würde ich so nicht bestätigen.

00:06:40 Sprecher 3

Also, erfahrungsgemäß ist man nahezu gleich schnell, wenn man etwas modelliert oder etwas von Hand programmiert.

00:06:50 Sprecher 3

Die Vorteile sehe ich da viel eher dann, wenn es ins Lifecycle geht, sprich, wenn man Anpassungen machen muss am Code,

00:07:00 Sprecher 3

weil man da, wie es sich zeigte, viel effizienter ist.

00:07:04 Sprecher 3

Vor allem der Grund, so wie ich es sehe, liegt darin, dass man durch eine grafische Modellierung eine sehr wertvolle Dokumentation erhält.

00:07:16 Sprecher 3

Das heißt, die Software ist eigentlich sehr gut dokumentiert, grafisch oder normalerweise dokumentiert man Software ja genau grafisch.

00:07:26 Sprecher 3

Und wenn die Dokumentation eigentlich das Gleiche ist wie der Code sozusagen und das immer in Sync ist, ist das ein Riesenvorurteil.

00:07:37 Sprecher 2

Ah, OK, das ist für mich also neue Erkenntnis.

00:07:39 Sprecher 2

Also, es war bei Ihnen jetzt gar nicht der Punkt, dass Sie Arbeit beim Codieren sparen wollen, sondern letztlich eigentlich beim Dokumentieren und beim Synchronhalten von Dokumentation, sprich Modell und dem Code.

00:07:52 Sprecher 2

Das ist schön, sehr schön zu hören.

00:07:55 Sprecher 2

Ich hatte auch schon viel, sehr viel Code generiert und bin genau an dieser Stelle eigentlich ins gegenteilige Problem reingelaufen, nämlich einen generierten Code synchron zu halten, weil es oft schwierig war, Geschäftslogik zu beschreiben.

00:08:11 Sprecher 2

Wie bekommen Sie das hin, dass Sie den Code synchron halten können oder die Frage anders formuliert?

00:08:17 Sprecher 2

Sind Sie in der Lage, im Modell alles

00:08:20 Sprecher 2

zu beschreiben, so dass es überhaupt keine Notwendigkeit mehr gibt, in den generierten Code einzugreifen.

00:08:25 Sprecher 3

Genau, das ist so bei uns.

00:08:27 Sprecher 3

Also, wir machen das so, dass wir eigentlich den generierten Code nicht mehr ändern.

00:08:33 Sprecher 3

Also, wir generieren immer nur in eine Richtung, niemals zurück.

00:08:38 Sprecher 3

Und ja, das benötigt oder das bedingt natürlich, dass man sämtliche Applikationslogik auch visuell oder im Modell beschreiben kann.

00:08:49 Sprecher 3

Ansonsten wird es natürlich schwierig.

00:08:52 Sprecher 2

Könnten Sie uns ein Beispiel für so ein Modellelement nennen?

00:08:57 Sprecher 2

Also, auf was für Notationselementen besteht es, damit wir uns vorstellen können, was man damit ausdrücken kann?

00:09:03 Sprecher 3

Da gibt es ganz viel verschiedene Modellsprachen, wie man was beschreiben kann.

00:09:10 Sprecher 3

Also, das können Filterelemente sein, die wir da reinziehen können.

00:09:16 Sprecher 3

Aber das geht wirklich so weit, dass wir Auditionen

00:09:19 Sprecher 3

grafisch beschreiben können.

00:09:21 Sprecher 3

Wir können diverse andere mathematische Operationen beschreiben.

00:09:26 Sprecher 3

Dann gibt es auch die Möglichkeit, dass wir Zustandsdiagramme, ja, wie man scannt von UML, beschreiben können.

00:09:34 Sprecher 3

Dann mit einer Skriptsprache direkt im Modell die Logik definieren können und dann aus dem ganzen Konstrukt den Code entsprechend generieren.

00:09:46 Sprecher 3

Also insofern kann man wirklich

00:09:48 Sprecher 3

auch sehr Low Level Operationen durchführen.

00:09:52 Sprecher 3

Was aber nicht geht, ist dann direkte Zugriffe auf Hardware und da braucht es immer eine Schnittstelle zwischen generiertem Code und handgeschriebenen Code.

00:10:01 Sprecher 3

Das ist auch etwas, das wir typischerweise so machen.

00:10:05 Sprecher 3

Also, Projekte generieren wir eigentlich nie ganzheitlich, sondern es ist immer noch ein gewisser Teil mit dabei, der handgeschrieben ist.

00:10:15 Sprecher 3

wichtig ist da einfach, dass man da klare Schnittstellen definiert und einhaltet, dass man wirklich den generierten Code nicht mehr anpassen muss, sondern dass dieser Code dann einfach kommuniziert mit anderen handgeschriebenen Code, welcher dann entsprechend auf die Hardware zugreift.

00:10:35 Sprecher 2

Mhm, Sie haben gerade angedeutet, dass sozusagen auch wenn die Notationselemente nicht ausreichend, Sie auch die Möglichkeiten haben, mit der Skriptsprache noch zu ergänzen,

00:10:45 Sprecher 2

Ist das eine eigenentwickelte Skriptsprache oder haben Sie sich darauf eine gesetzt, die es bereits gibt, wie keine Ahnung, JavaScript zum Beispiel.

00:10:52 Sprecher 3

Ja, in unserem Fall ist das jetzt im Bereich von Simulink und und Matlab ist es einfach die das M.

00:11:01 Sprecher 3

Skript, das man da verwenden kann und aus diesem M.

00:11:05 Sprecher 3

Skript wird dann entsprechend auch Co generiert.

00:11:09 Sprecher 3

Aber man hat auch die Möglichkeit, direkt in C.

00:11:13 Sprecher 3

Syntax was reinzuschreiben, das dann entsprechend übernommen.

00:11:19 Sprecher 2

Wird.

00:11:19 Sprecher 2

Wir haben vorhin bereits schon über das Thema Dokumentation gesprochen gehabt und es ist ja eine der wesentlichen regulatorischen Anforderungen an Medizinprodukte oder auch an die Entwicklung von medizinischer Software.

00:11:32 Sprecher 2

Jetzt haben wir auch regulatorische Anforderungen an Code-Generatoren.

00:11:38 Sprecher 2

Welche würden Sie da

00:11:39 Sprecher 2

als besonders relevant sehen und wie haben Sie diese Anforderung erfüllt?

00:11:44 Sprecher 3

Also, am wichtigsten für uns ist sicher oder im wichtigsten im Zusammenhang mit Code-Generierung ist sicher die 13485 und die 62304, die uns da gewisse Vorgaben gibt.

00:12:00 Sprecher 3

Jetzt im Bereich der 13485 ist es sicher die Anforderung, die die Norm stellt, dass wir Tools

00:12:09 Sprecher 3

Und Werkzeuge, also ein Co-Generator, validieren, sofern er beteiligt ist am Entwicklungsprozess für das Medizingerät.

00:12:19 Sprecher 2

Und das haben Sie dann offensichtlich auch getan.

00:12:22 Sprecher 2

Jetzt haben Sie auch die 62 304 nochmal genannt.

00:12:25 Sprecher 2

Das ist ja eigentlich eine Norm, die sich um Software kümmert, die ins Medizinprodukt reinkommt und das ist jetzt bei einem Code-Generator nicht der Fall.

00:12:34 Sprecher 2

Wie behandeln Sie aus 62 304 Sicht den generierten Code?

00:12:39 Sprecher 2

Nutzen Sie den als Soup oder sagen Sie, nee, den behandeln wir eigentlich wie selber geschriebenen Code und wenden dann quasi zusätzlich nochmal alle Elemente der Qualitätssicherung, die die 62 304 vorsieht, darauf an?

00:12:54 Sprecher 2

Welche der bei den Versionen nutzen Sie?

00:12:57 Sprecher 3

Ja, also grundsätzlich als Soup, denke ich, wird es relativ schwierig werden, weil gemäß Definition ist eine Soup etwas, das

00:13:08 Sprecher 3

nicht für das Produkt ursprünglich entwickelt wurde, um darin integriert zu werden, sondern ja, eine SUP ist typischerweise eine Library oder sonst was.

00:13:22 Sprecher 3

Klar, kann man da mit viel Kreativität versuchen zu argumentieren, aber ich bezweifle, dass das ein Auditor akzeptieren wird.

00:13:31 Sprecher 3

Von dem her unterwerfen wir dem generierten Code

00:13:37 Sprecher 3

Ja, ganz normal der 62304, wie sie von Hand geschrieben wäre, und führen da exakt die gleichen Tätigkeiten durch.

00:13:47 Sprecher 3

Sprich, wir führen da eine statische Code-Analyse durch mit entsprechenden Metriken und wir machen dynamischen Tests der einzelnen Komponenten und wir stellen auch entsprechende Rückverfolgbarkeit sicher zwischen Anforderungen

00:14:05 Sprecher 3

und Code.

00:14:06 Sprecher 3

Es ist aber schon so, dass man das nicht unbedingt machen muss.

00:14:10 Sprecher 3

Also im Beispiel vom Compiler, das ja auch ein Code-Generator ist, dort ist es nicht so, dass wir dort den Maschinencode nochmals prüfen und sämtliche Checks durchführen.

00:14:26 Sprecher 3

Aber das ist so, dass wir das anders argumentieren können.

00:14:32 Sprecher 3

Das heißt, beim

00:14:34 Sprecher 3

Bei einem Compiler kann man sich darauf berufen, dass die Wahrscheinlichkeit sehr, sehr, sehr gering ist, dass dieser uns ein Fehler einschleust in die Applikation, weil dieser tausendfach verwendet wurde.

00:14:49 Sprecher 3

und Somit können wir da eine kleine Validierung machen, der des Compilers und damit ist gut.

00:14:55 Sprecher 3

Hingegen bei einem Code-Generator, vor allem bei einem Code-Generator, der selbst geschrieben wurde,

00:15:05 Sprecher 3

kann man das auf Blackblock Sicht beinahe nicht wäre genügend verifizieren, weil man da in das System Code Generator nicht rein sieht und von dem her wird es da schon schwierig zu argumentieren.

00:15:20 Sprecher 3

Ja, der wird uns keine Fehler mit einschleusen, weil es ja nicht so tausendfach oder millionenfach verwendet wird wie ein Compiler und von dem her.

00:15:33 Sprecher 3

bleibt uns schlussendlich nur noch die Möglichkeit, wirklich das Generat entsprechend zu prüfen und sicherzustellen, dass die Funktion, für das es da ist, auch tatsächlich erfüllt.

00:15:44 Sprecher 2

Ja, vielen Dank.

00:15:45 Sprecher 2

Also vielleicht 2 Gedanken dazu und noch eine Frage.

00:15:50 Sprecher 2

Erster Gedanke, also bei Sub

00:15:53 Sprecher 2

kann man schon auch eben nicht nur Bibliotheken mit reinnehmen, sondern auch eigenentwickelten Code, für den es keine qualitätssichernden Maßnahmen gab.

00:16:00 Sprecher 2

Das ist nämlich genau, was SUP von O.T.S.

00:16:02 Sprecher 2

unterscheidet.

00:16:04 Sprecher 2

Zweite Gedanke, das was sie machen, ist ‚n Risk-based Approach und das ist eigentlich genau das, was man sich auch wünscht, dass man nämlich drüber nachdenkt, mit welcher Wahrscheinlichkeit kann es da ein Fehler noch mal durchrutschen.

00:16:18 Sprecher 2

Und so wie ich das gehört hab, sind sie da auf auf der sicheren Seite, auf die sie sich hinbewegen,

00:16:23 Sprecher 2

dass sie nämlich sowohl den Code-Generator validieren und zwar nicht nur als Blackbox, wie ich sie gerade verstanden hab, und dass sie zusätzlich noch das erzeugte Kompilat, also den Quellcode, den ihr Code-Generator ausspuckt, auch noch mal einer vollständigen Qualitätssicherung unterwirft, so wie sie das für den eigenen Code gemacht haben.

00:16:43 Sprecher 2

Also sozusagen doppelt gemäht hält hier besser.

00:16:45 Sprecher 2

Eine Frage noch dazu, sie haben gerade gesagt, dass sie als Teil dieser

00:16:51 Sprecher 2

Überprüfung, also der Qualitätssicherung des generierten Codes, auch eine statische Code-Analyse machen.

00:16:57 Sprecher 2

Wieso lässt sich sowas nicht bereits auf der Ebene des Code-Generators überprüfen?

00:17:03 Sprecher 2

Oder anders gefragt, wie kann es sein, dass der Code-Generator einen Code generiert, der nicht den Vorgaben einer statischen Code-Analyse erfüllt?

00:17:13 Sprecher 2

Wie könnte sowas passieren?

00:17:15 Sprecher 3

Also, das kann darin passieren, dass halt

00:17:20 Sprecher 3

ein Fehler, in dem Code-Generator vorhanden ist, der irgendwas generiert, das nicht zulässig ist, wie beispielsweise eine Division durch 0 oder ein Nullpointer-Zugriff oder was auch immer.

00:17:32 Sprecher 3

Und von dem her, ja, wenn man da keine Validierung hat vom Code-Generator und wirklich auch solche Low-Level-Instruktionen ausführen kann im Modell, dann

00:17:47 Sprecher 3

ja, ist man davon nicht gefeit.

00:17:49 Sprecher 3

Also beispielsweise kann man sehr wohl auch auf Modellebene Sachen definieren, die nicht gut sind, also die gefährlich sind, weil das Problem ist ein bisschen das, dass viele Co-Generator haben da schon explizite Prüfungen, die sowas jeweils prüfen.

00:18:13 Sprecher 3

Aber was halt dann

00:18:15 Sprecher 3

daraus resultiert ist möglicherweise ineffizienter Code.

00:18:19 Sprecher 3

Das heißt, wenn der Code-Generator davon ausgeht, dass der oder wenn der Code-Generator den Benutzer von sich selbst schützen möchte auf irgendeine Art resultiert, das sind sehr viele zusätzliche Prüfungen, die damit generiert werden.

00:18:35 Sprecher 3

Und wenn man ja nicht von endlos Leistung des Prozessors ausgehen kann, ist man

00:18:44 Sprecher 3

je nachdem angewiesen, solche Prüfungen eben nicht dazu generieren lassen.

00:18:49 Sprecher 2

Verstehe, das heißt, es war quasi ein Trade-off, an welche Stelle Sie diese letztlichen Prüfungen setzen und sind zum Ergebnis gekommen, dass es manchmal eleganter sein kann, das eben nicht in den Generator oder in die Evaluation des Modells einzubauen, sondern letztlich dann in zum Beispiel eine statische Codeprüfung des generierten Codes.

00:19:10 Sprecher 2

O.

00:19:11 Sprecher 2

K.,

00:19:11 Sprecher 2

Ja, da merkt man schon, wie viel Erfahrung sowas notwendig ist, um nicht nur zu modellieren, sondern auch entsprechende Generatoren zu kaufen oder zu bauen.

00:19:22 Sprecher 2

Und Sie haben ja da offensichtlich sehr viel Erfahrung.

00:19:26 Sprecher 2

Bei welchen Projekten könnten Sie jetzt anderen Firmen, die jetzt in in ähnlicher Situation vielleicht sind, besonders gut helfen?

00:19:34 Sprecher 2

Also, wir würden auch Ihre Kontaktdaten verlinken.

00:19:38 Sprecher 2

Bei wem macht es besonders Sinn, dass er sich oder sie sich bei Ihnen meldet?

00:19:41 Sprecher 3

Also grundsätzlich vor allem, wenn man im Bereich von Embedded Geräte was entwickeln lassen würde.

00:19:50 Sprecher 3

Also wir verwenden unsere Co-Generator hauptsächlich für Embedded Geräte.

00:19:55 Sprecher 3

Dort gehören sie uns bei uns ja zum Standard, außer der Kunde möchte das explizit nicht.

00:20:02 Sprecher 3

Aber wir sind sehr stark von den Vorteilen überzogen.

00:20:07 Sprecher 3

was uns diese Code-Generatoren bringt.

00:20:10 Sprecher 3

Was wir, wie ich bereits auch schon erwähnt habe, auch noch haben, ist dieses eigene Tool, das auch Code generieren kann.

00:20:21 Sprecher 3

Und dieses Tool haben wir beschlossen, da es sich ja sehr stark bewerte in unserem Entwicklungsprozess.

00:20:30 Sprecher 3

Dieses Tool haben wir beschlossen, dass wir dieses Kommerzial analysieren.

00:20:35 Sprecher 3

unter der Firma Dataflow A.

00:20:38 Sprecher 3

G.

00:20:38 Sprecher 3

Ja, kann man da mal reinschauen und prüfen, ob das vielleicht etwas ist.

00:20:45 Sprecher 2

Das heißt, Sie haben sogar 2 Möglichkeiten, andere zu unterstützen und von ihren Erfahrungen profitieren zu lassen.

00:20:51 Sprecher 2

Nämlich einmal, indem Sie wirklich beim Engineering direkt helfen und zum anderen, indem Sie das Tool, das sich bei Ihnen offensichtlich schon bewährt hat, auch anderen dann zur Verfügung stellen.

00:21:02 Sprecher 2

Ja, großartig.

00:21:03 Sprecher 2

Da danke ich ganz herzlich.

00:21:05 Sprecher 2

Herr van der Steyr hat Riesenspaß gemacht.

00:21:08 Sprecher 2

Für alle, die noch mehr wissen wollen zu dem Thema, die dürfen sich gern vertrauensvoll annehmen.

00:21:13 Sprecher 2

Ansonsten bleibt mir noch mal ganz herzlichen Dank zu sagen und bis nächste Woche.