Schwingungserreger für Strukturtests dynamisch belasteter Bauteile

Ferdinand Friedrich, Dr.-Ing. Thomas Anderl
IABG – Industrieanlagen-Betriebsgesellschaft mbH, Ottobrunn

Andreas Pfichner
APSysteme GmbH, Unterhaching

Kurzfassung

In diesem Beitrag wird die Realisierung einer Prüfanlage zur dynamischen Belastung von komplexen Strukturbauteilen vorgestellt. Beispielsweise erfolgt der Nachweis der Betriebsfestigkeit von Rotorblättern für die Zertifizierung von Windenergieanlagen in der Regel im Dauerschwingversuch über mehrere Millionen Lastspiele. Mit einem von der IABG entwickelten Verfahren kann die Testdauer erheblich reduziert und die Testdurchführung wesentlich flexibler und robuster ausgeführt werden.

Durch eine übersichtlich gestaltete und einfach gehaltene LabVIEW-Architektur können anspruchsvolle cRIO-Applikationen realisiert werden. Die Applikationen stellen bei bestmöglicher Performance den Kompromiss zwischen Sicherheit und effizienzoptimierten Entwicklungsprozessen dar. Der Einsatz einer State Machine als Maschinensteuerung gekapselter Aktorarchitekturen sowie die Verwendung von LabVIEW-Bibliotheken (Libraries) führt zu Modularität und gesteigerter Akzeptanz in heterogenen/interdisziplinären Entwicklerteams.

Der Beitrag fasst die modulare und performante Einbettung der Prüfstandsfunktionen in ein CompactRIO-Framework zusammen. Das Projekt wurde bezüglich der Softwareentwicklung in Zusammenarbeit mit einem LabVIEW Consultant realisiert.

Abstract

Here we introduce a test facility concept for dynamic testing of complex structures. The certification of wind turbines requires at least a few million stress cycles to prove the engineering strength by dynamic fatigue tests of rotor blades. IABG has developed a testing process significantly reducing the testing time and performing in a highly flexible and robust manner.

Challenging cRIO applications can be realized with our clear and simple LabVIEW architecture. With optimal performance these applications are able to solve the compromise between safety and even more efficient development processes. The use of a state machine for machine control increases the encapsulated actuator architectures as well as the LabVIEW libraries modularity and the acceptance in heterogeneous/interdisciplinary teams of developers.

This article summarizes the embedding of modular functions in a CompactRIO framework. Regarding the software development, this project was realized in cooperation with a LabVIEW consultant.

Prüfsystem für effiziente Rotorblatt-Tests

Es sollte eine applizierbare Prüfanlage entwickelt werden, mit der z. B. Ermüdungstests an großen Strukturen (**Bild 1**) mit Resonanzeffekten durchgeführt werden können.

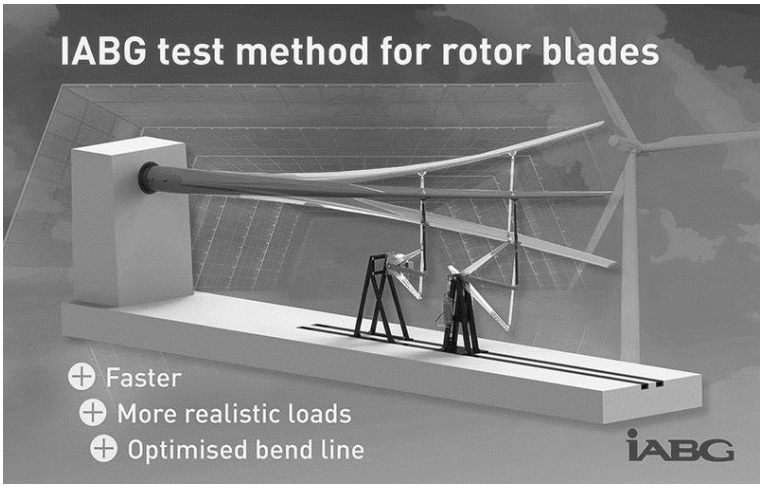

Bild 1: Schematische Darstellung der Prüfanlage

Dabei sollte abhängig von Steifigkeit und Abmaß der Struktur ein variabler Krafteinleitungspunkt gewählt werden können. Um die Struktur in ihrer Resonanz zu testen, ist ein sinusförmiges Stellsignal notwendig. Zusätzlich bestand die Anforderung, dass sowohl die Amplitude als auch die Frequenz des sinusförmigen Stellsignals wahlweise über ein GUI Interface, ein Bedienpult oder kundenspezifische Systeme vorzugeben sind. Des Weiteren sollte die Prüfanlage u. a.

- einen Einrichtbetrieb zur Kopplung des Krafteinleitungspunkts und der Struktur
- die Sollwinkelvorgaben über Handbedienelemente
- taktgenaue und -treue Auskopplung des Stellsignals (Timed Loop)
- eine Momentregelung
- Visualisierung, Aufzeichnung und Verwaltung von Parametern und Messwerten

ermöglichen.

Modulares Automatisierungskonzept auf Basis von CompactRIO und LabVIEW

Das Konzept (**Bild 2**) zeigt, dass einzelne Softwaremodule hardwarenah appliziert wurden. Zusätzlich geht hieraus hervor, dass die Einsatzvielfalt des CompactRIO zur Bündelung einzelner Softwarefunktionen genutzt wird. Somit kann für zukünftig entwickelte Softwaremodule ein Interface (via DMA bzw. via Networkstreams) geboten werden.

Bevor die Prüfanlage in den Automatik- bzw. Regelmodus übergeben werden kann, ist eine gezielte Anregung und Identifikation der Struktur im Prüfaufbau notwendig. Dazu wird z. B. die Frequenz des anregenden Sinussignals an die im Vorfeld berechnete Eigenfrequenz angenähert und angeregt. Das Motormoment wird hierfür über eine Koppelstange und spezielle Schnittstellen als frontale Kraft in das Bauteil eingeleitet. Zusätzlich kann die Amplitude der

Kraftanregung auf dem GUI des CompactRIO variabel eingestellt werden. Im Automatikmodus wird der Regelkreis geschlossen. Die wesentliche Regelstrategie ist dabei eine Verschiebung des Betriebspunkts in den aufsteigenden Ast der Resonanzkurve. Durch Verstellung der Schwingfrequenz bei gleichbleibender Anregungshöhe erfolgt bei Frequenzerhöhung automatisch auch eine Erhöhung der Resonanz und somit der Amplitude. Bei Frequenzreduzierung läuft der beschriebene Ablauf konträr ab, sodass sich die Schwingamplitude reduziert.

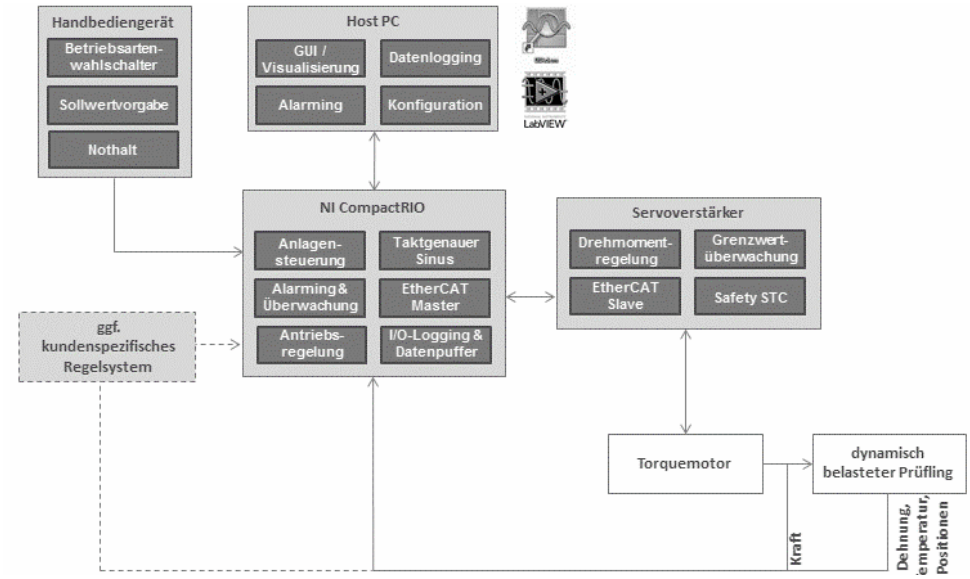

Bild 2: Funktionsarchitektur des Schwingungserregers

LabVIEW Actor Framework vs. State Machine

Bei RT-Steuerungs- und Regelungsaufgaben, die aufgrund geringer Iterations-Periodenzeiten paralleler Regelschleifen eine hohe CPU-Last des cRIO Target verursachen, kommt es auf hohe Performance an. Da durch bestimmte *consuming design patterns*, wie z. B. das bekannte LabVIEW Actor Framework, Performance-Verluste verursacht werden können, sind diese aufgrund der hohen Anforderungen nicht praktikabel. Durch die Vermeidung von *shared resources*, die Verwendung von Timed Loops und den Grundsatz „Keep it simple" kann auf RT-Seite die notwendige Performance erzielt werden.

Neben der hohen Anforderungen an die Leistungsfähigkeit werden an die vorliegende Applikation auch hohe Sicherheitsanforderungen gestellt. Das wohl bekannteste und zugleich sicherste Designpattern für Maschinensteuerungen ist die State Machine.

Komplexe Entwicklungsprozesse sowie die Berücksichtigung von Effizienz und Kostenersparnis erfordern insbesondere bei der Anlagenentwicklung modulare und skalierbare Architekturen. Diese können durch eine einfach gehaltene State Machine nicht unmittelbar erzielt werden. Aus diesem Grund wird ein aktorbasierter Ansatz gewählt, sodass die Entwicklung modular und effizient in interdisziplinären Teams erfolgen kann.

Es stellt sich nun die Frage, wie die Anforderungen an hohe Performance und höchste Ablaufsicherheit erfüllt werden können. Zusätzlich müssen diese Kriterien auch nach Code-Erweiterungen weiterhin erfüllt werden.

State Machine und Aktoren

Die Antwort auf die oben gestellte Frage kann wie üblich durch einen Kompromiss gegeben werden. Der vorliegende Anwendungsfall erlaubt den Einsatz eines LabVIEW Actor Framework auf der einen Seite und einer einfach gehaltenen State Machine andererseits.

Für den sicheren Einsatz einer State Machine mit Aktoren muss auf eine geregelte Kommunikation geachtet werden. Dabei kommunizieren die Aktoren, im Folgenden als Module bezeichnet, ausschließlich mit der State Machine. Es existiert also auf dem RT Target kein Kommunikationskanal zwischen den einzelnen Modulen. Des Weiteren kommunizieren die Module ausschließlich über die RT State Machine mit sich selbst. Einzig das User Interface (GUI) stellt bei der Kommunikation eine Ausnahme dar, welches auf der Host-Ebene keine unmittelbare Gefahr für die RT-Maschinensteuerung darstellt. Hier werden Moduldaten direkt, ohne die State Machine zu durchlaufen, an das GUI gesendet. Dies ist beispielsweise für die Anzeige der aktuellen Messwerte notwendig.

Die State Machine, die auf dem RT Target ausgeführt wird, kommuniziert mit allen parallelen Tasks und Modulen (**Bild 3**).

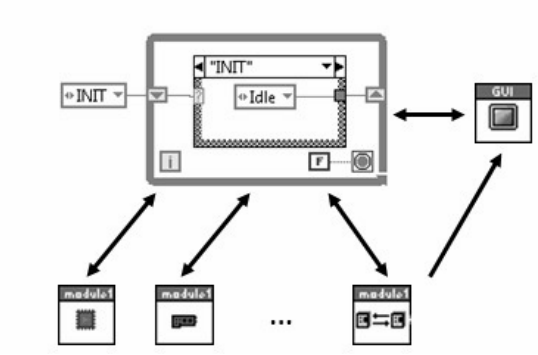

Bild 3: Kommunikationsmodell State Machine ⟺ Aktoren/Module

Auf diese einfache Weise und unter konsequenter Einhaltung von Kommunikationsparadigmen durch das Softwareentwicklungsteam können Aufgaben modular entwickelt und debugged werden. Gleichzeitig bleiben alle Vorteile der klassischen State Machine bestehen – es werden also z. B. Race Conditions ausgeschlossen und es kann nicht zu unvorhergesehenen Zuständen kommen.

Datenaustausch zwischen Host und RT Target

Der Datenaustausch zwischen Host- und RT-Ebene findet standardmäßig mit den LabVIEW Network Streams statt. Dabei hat es sich bewährt, *alle* Datenpakete (z. B. Messages an Module, aktuelle Daten) durch einen standardisierten Message Cluster zu verschicken (**Bild 4**).

Die vorliegende Architektur verwendet lediglich einen Network Stream für den Daten-Message-Upstream und einen für den Message Downstream.

Da im vorliegenden Fall die Vorteile des Einsatzes von *Strings* überwiegen, wird auf die Verwendung von *Enums* verzichtet. Strings haben u. a. den Vorteil, dass Namen von Messages aufwandsreduziert angepasst und Änderungen/Erweiterungen innerhalb eines Moduls eingearbeitet werden können. Bei Verwendung eines Versionsverwaltungstools (z. B. SVN)

kann ein Versionsupdate der gesamten Applikation vermieden und ein bestimmter Entwicklungsschritt modular entkoppelt werden.

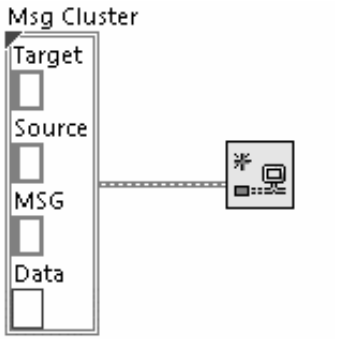

Bild 4: Network Streams mit standardisiertem Message Cluster für den HOST-RT-Up- und -Downstream

Voraussetzung für den effizienten Einsatz von stringbasierten Message-Clustern bei einem Actor Framework ist, dass für das dadurch adressierte Modul eine API durch den Entwickler bereitgestellt wird. Für den Fall, dass sich im Lauf der Entwicklung Änderungen an den einzelnen Modulen (z. B. Änderung eines Modul-State-Namens), die über einen MSG String adressiert werden, ergeben, so sind selbstverständlich eine Aktualisierung und erneute Überprüfung der API erforderlich. Auch wenn Enum Typedefs in großen Entwicklungszusammenhängen eher hinderlich sind, ist ihr Einsatz in kleineren Projekten umso hilfreicher.

Modul-, Aktoren-Architektur und API

Der prinzipelle schematische Aufbau eines Moduls (**Bild 5**) ist charakterisiert durch eine *named queue*. Vergleichbar zum Aufbau der o. g. Streaming Message besitzt auch das Modul einen Message Cluster, worüber die Adressierung ermöglicht wird.

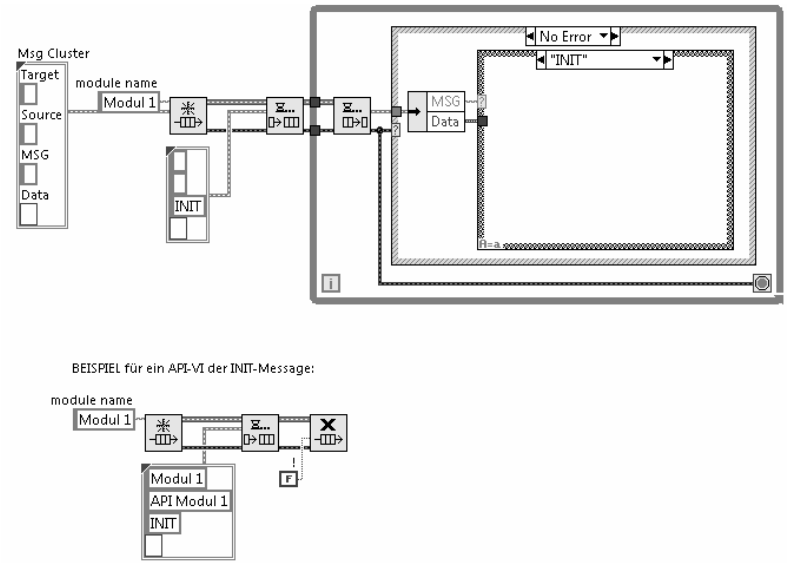

Bild 5: Prinzip einer Modularchitektur und zugehöriger API

Der große Vorteil in der Verwendung des identischen Message Cluster im Network Stream und im Actor Queue liegt darin, dass Messages direkt aus dem RT-Modul an die Host-Ebene gestreamt werden können. Hinzu kommt, dass die Modularchitekturen sowohl auf RT- als auch auf Host-Ebene identisch ausgeführt sind. Durch diese Architekturausführung ist zum einen ein plattformübergreifendes Programmieren möglich und auch das Debugging vieler RT-Prozesse wird auf der Host-Ebene ermöglicht. Damit wird die Effizienz in der Entwicklung weiter gesteigert.

Durch die in der Modul-API enthaltene Case Structure mit der Option von *case-insensitive matches* sollen *syntax errors* in Bezug auf die Groß-/Kleinschreibung vermieden werden. Durch das Zusammenfassen der Modul-API-VI in LabVIEW-Bibliotheken kann eine gute Alternative zur Objektorientierung geschaffen werden.

Zusammenfassung

Die anfänglich hohen Anforderungen an die Architektur bezüglich Skalierbarkeit und Modularität konnten durch die leistungsstarke Kombination aus cRIO und LabVIEW erfüllt werden. Des Weiteren wurden Architekturmodule erstellt, welche die Einbindung von externen kundenspezifischen Regelsystemen ermöglichten. Durch den modularen Ansatz konnten Funktionen frühzeitig einzeln getestet werden, was in entscheidenden Projektabschnitten zu einer erheblichen Zeitersparnis führte. Zudem wurden durch die performante Architektur dem interdisziplinären Entwicklerteam Innovationsspielräume geboten.

Aufgrund der vielen Vorteile wird dieser Automatisierungsansatz von der IABG als modulare Plattform für System- und HiL-Prüfstände und Anlagen eingesetzt.