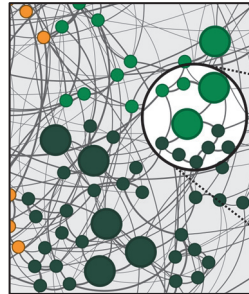


Überprüfung von Schnittstellenkompatibilitäten unter Anwendung von Wissensgraphen



Stephan, R.; Inkermann, D.

Um die Funktionalitäten komplexer mechatronischer Produkte gewährleisten zu können, müssen unter anderem die Schnittstellen zwischen Subsystemen vor der domänenspezifischen Entwicklung ausreichend definiert werden. Dies ist oftmals nur mithilfe von Wissen aus den jeweiligen Entwicklungsabteilungen möglich, weshalb zunächst eine Festlegung der Architektur und erst im Anschluss eine detaillierte Schnittstellendefinition erfolgt (Bottom-Up-Ansatz). Dies führt häufig zu Schnittstelleninkompatibilitäten zwischen Subsystemen, die mit Hilfe von Wissensgraphen überprüft werden können. Ein Wissensgraph ist eine strukturierte Darstellung von Daten als Knoten (grüne Kreise im Titelbild) und Beziehungen zwischen Daten als Kanten (graue Linien im Titelbild), die maschinell analysiert werden können.

In order to ensure the functionality of complex mechatronic products, the interfaces between subsystems must be sufficiently defined prior to domain-specific development. This is often only possible with the help of knowledge from the respective development departments, which is why the architecture is first defined and only then is a detailed interface definition carried out (bottom-up approach). This leads to interface incompatibilities between subsystems, which can be checked using knowledge graphs. A knowledge graph is a structured representation of data as nodes (green circles in the title image) and relationships between data as edges (gray lines in the title image) that can be analyzed by computers.

Einführung und Problemstellung

In der Literatur wird Systems Engineering meist Top-down-Vorgehen beschrieben /1/. Im Zuge der Architekturdefinition als eine Hauptaktivität im Systems Engineering /2/, wird das System of Interest (Sol) in unterschiedliche Subsysteme unterteilt, welche im weiteren Verlauf ausdetailliert und parallel entwickelt werden, wie in Abbildung 1 zu erkennen. Dies erfolgt, um die Komplexität mechatronischer Produkte besser handhaben zu können und stellt sicher, dass Verantwortlichkeiten besser zugeordnet werden können. Da die Subsysteme miteinander interagieren, müssen eindeutige Schnittstellen definiert werden. Eine Schnittstelle ist definiert als eine gemeinsame Grenze zwischen zwei Funktionseinheiten, definiert durch verschiedene Merkmale in Bezug auf die Funktionen, den physikalischen Signalaustausch und andere Eigenschaften (ISO/IEC 199) /3/. Bei der Fahrzeugentwicklung werden beispielsweise sieben Subsysteme definiert, die das

Fahrzeug bilden. Zwischen dem Fahrsystem und dem Energiesystem, zwei Teilsysteme des Fahrzeugs, bestehen unter anderem eine Vielzahl von Verbindungen, um die Funktion der Rekuperation zu realisieren. Dabei werden Informationen, beispielsweise über den Batteriezustand oder die Ladegeschwindigkeit, sowie Energie zwischen den Subsystemen ausgetauscht. Bei der Definition von Schnittstellen wird genau festgelegt, welche Anzahl, welche Art und welche Richtung des Austauschs vorliegen. Allgemein kann zwischen einem Austausch von Material, Energie, Information sowie einer physischen Verbindung unterschieden werden /4/. Sind die Schnittstellen nicht korrekt definiert, führt dies zu Problemen bei der Integration der entwickelten Subsysteme /5/. Dies kann sich in physischen Inkompatibilitäten, zum Beispiel in nicht aufeinander abgestimmten Bohrungen, oder in unterschiedlich verwendeten Datenformaten für den Informationsaustausch äußern. Somit ist es essentiell, Schnittstellen frühzeitig zu definieren, da so eine reibungslose Integration nach der Entwicklung garantiert werden kann /6/. Die Festlegung von Schnittstellen im Zuge der Architekturdefinition ist zudem wichtig, weil so sichergestellt werden kann, dass das Gesamtsystem mit der Umwelt sowie mit Subsystemen korrekt verbunden ist /7/.

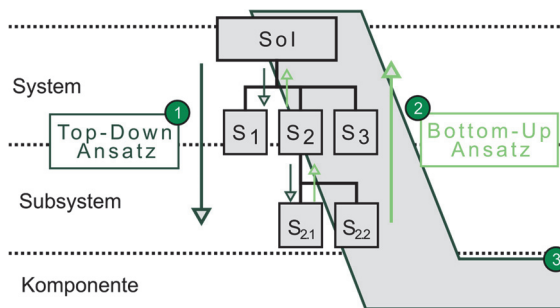


Abbildung 1: Vorgehensweisen im Systems Engineering Top-Down vs. Bottom-Up

Zu Beginn der Architekturdefinition werden die Schnittstellen festgelegt. Um diese detailliert zu beschreiben, bedarf es Expertenwissen aus den jeweiligen Entwicklungsabteilungen und -domänen. Somit wird die Architektur der Subsysteme parallel entwickelt (Definition von Funktionen und Komponenten auf unterster Detaillierungsebene), und genaue Eigenschaften der Schnittstellen werden erst spät mit anderen Entwicklungsabteilungen abgestimmt. Als Folge der Anwendung dieses Bottom-Up-Vorgehens (siehe Nr. 2 in Abbildung 1) liegen häufig Schnittstelleninkompatibilitäten zwischen Subsystemen vor. Werden diese erst in nachfolgenden Aktivitäten, zum Beispiel im Systementwurf, festgestellt, entstehen Nacharbeiten und Änderungen an der Architektur, die mit Kosten verbunden sind.

Model-based Systems Engineering (MBSE) bezeichnet die Anwendung formalisierter Modelle im Zuge des Systems Engineerings. Neben anderen Entwicklungsaktivitäten kann das MBSE auch für die Architekturdefinition und somit für die Schnittstellendefinition verwendet werden. Der Vorteil dabei ist, dass die Schnittstellen in formalisierter Form vorliegen und damit das Potenzial für einen

automatischen Abgleich bieten, um die Kompatibilität zu überprüfen. Um beliebig viele Schnittstellen mit einer Vielzahl von Parametern abgleichen zu können, benötigt es einen automatischen und robusten Ansatz, der mithilfe von Wissensgraphen realisiert werden kann.

Zielstellung des Beitrags

Ziel des Beitrages ist es, eine strukturierte Übersicht relevanter Schnittstelleneigenschaften zu erstellen, die im Zuge des MBSE für den Überprüfung auf mögliche Inkompatibilitäten verwendet werden können. Folgende Fragestellungen werden dazu in diesem Beitrag adressiert:

- Welche Verbindungstypen zwischen Subsystemen können durch Schnittstellen definiert werden?
- Welche in der Architekturdefinition festgelegten Schnittstelleneigenschaften sind für eine Überprüfung auf Inkompatibilitäten relevant?

Um diese Forschungsfragen zu adressieren, werden zunächst die Grundlagen verschiedener Verbindungstypen zwischen Subsystemen aufgezeigt. Darauf basierend werden in der Architekturentwicklung relevante Schnittstelleneigenschaften definiert. Anschließend werden die SysML-Elemente für einen Schnittstellenabgleich festgelegt. Der Beitrag endet mit einer Zusammenfassung sowie einem Ausblick auf die Anwendung von Wissensgraphen für den Schnittstellenabgleich.

Schnittstelleneigenschaften für den Schnittstellenabgleich

Um einen möglichst vollständigen Abgleich der Schnittstelleneigenschaften zu ermöglichen, werden zunächst die grundsätzlichen Verbindungstypen zwischen Subsystemen betrachtet. Unter Berücksichtigung jener lassen sich im Anschluss die im Zuge der Schnittstellendefinition relevanten Eigenschaften festlegen. Wilms et al. (2019) definierten auf Grundlage von Literatur grundlegende Verbindungstypen und assoziierte Begriffe, die in Tabelle 2 aufgelistet sind /8/. Daraus abgeleitet sind in der zweiten Spalte die zu modellierenden Eigenschaften im Zuge der Schnittstellendefinition. Diese Eigenschaften sind zunächst sehr abstrakt definiert und reichen nicht aus, um einen vollständigen Schnittstellenabgleich durchführen zu können. Obwohl von zwei Schnittstellen, zum Beispiel die Austauschart und die Richtung des Austauschs, korrekt modelliert sind, kann nicht darauf geschlossen werden, dass diese in der Integration korrekt miteinander interagieren, da weiterführende Eigenschaften wie das Austauschmedium fehlen. Daher wurden in Tabelle 2 auf Basis von Tabelle 1 die Eigenschaften definiert, die im Zuge der Schnittstellendefinition im MBSE detailliert werden müssen. Es zeigt sich, dass in der Definition von Schnittstellen unterschiedliche Abstraktionsebenen berücksichtigt werden müssen. Dabei kann grundsätzlich zwischen der Black-Box- und der White-Box-Perspektive unterschieden werden /8/. In der Black-Box-Perspektive werden

nur die Subsysteme und die Schnittstellen zwischen den Subsystemen modelliert, wobei in der White-Box-Perspektive die Sicht durch Komponenten, die die Schnittstellen umsetzen, erweitert wird.

Tabelle 1: Verbindungstypen nach /8/ und abgeleitete Eigenschaften für die Schnittstellenmodellierung im MBSE

Verbindungstyp nach /8/	Zu modellierender Schnittstellenaspekt
Materialaustausch	Richtung und Austausch von Material
Räumliche Beziehung oder physikalische Verbindung	Physische Verbindung
Übertragung von Lasten, Kräften oder Drehmomenten	Richtung und Austausch von Energie
Translatorische oder rotatorische Bewegung	Physische Verbindung, Richtung und Austausch von Energie
Vibrationen und Akustik	Richtung und Austausch von Energie
Magnetfeld	Richtung und Austausch von Energie
Thermische Beziehung	Richtung und Austausch von Energie und Material
Elektrisches oder elektromagnetisches Feld	Richtung und Austausch von Energie
Elektrische Erde	Richtung und Austausch von Energie
Elektrische Leistung	Richtung und Austausch von Energie
Informations- oder Steuersignale	Richtung und Austausch von Information

Schnittstellenmodellierung im MBSE

Nachdem grundsätzliche Eigenschaften der Schnittstellenmodellierung auf Basis von Verbindungstypen abgeleitet wurden, sollen im nächsten Schritt unter Einbezug weiterführender Literatur zur Schnittstellenmodellierung im MBSE relevante SysML-Elemente für einen Schnittstellenabgleich herausgearbeitet werden. Fosse et al. (2012) schlagen einen Ansatz zur modellbasierten Definition von Schnittstellen sowie relevante Eigenschaften von Schnittstellen vor /7/. Shames et al. (2016) entwickelten einen Ansatz zur Schnittstellenmodellierung, der auf unterschiedlichen Modellierungsebenen basiert. Hierbei wurde ebenfalls eine Übersicht von Eigenschaften erstellt, die für die Modellierung mit der Systems Modelling Language (SysML) im MBSE relevant sind. /3/. Basierend auf Tabelle 1 sowie den eingeführten Eigenschaften und assoziierten SysML-Elementen, ist in Tabelle 2 die Übersicht von SysML-Elementen für den Schnittstellenabgleich aufgezeigt.

Tabelle 2: Schnittstelleneigenschaften und SysML-Elemente, nach /3, 7, 8/

Schnittstellen-eigenschaft	Erläuterung	SysML Element
Verbindungstyp	Die Art der Verbindung (Physisch, Austausch)	Stereotyp Flow-/ ProxyPort
Art & Richtung des Austauschs	Austauschart (Material, Energie, Information) und Richtung des Austauschs	Flow zwischen Block Port & Property Port
Medium der Schnittstelle	Realisierung der Schnittstelle, z.B. als Kabel oder drahtlose Verbindung, Möglichkeit zur Ergänzung von Funktionen-	Stereotyp FullPort
Limitation der Schnittstelle	Limitationen des Austauschs z.B. eine maximale Anzahl an Drehmoment	Constraints
Nutzungsbedingungen & -szenarios	Funktionen, in denen die Schnittstelle, oder Teilfunktionen davon, Anwendung finden.	Tagged Values, Sequenzmodellierung

Beim Schnittstellenabgleich im Rahmen der Architekturdefinition werden die in der logischen Sicht festgelegten Eigenschaften verglichen. Entscheidend für diesen Vergleich sind der Verbindungstyp und die Verbindungsrichtung zwischen Block-Port und Property-Port sowie der Stereotyp des Ports, siehe Tabelle 2. Die zusätzliche Inbetrachtung des Port-Namen, des Freitexts zur Beschreibung des Ports und des Elementtyps, an welchem der Port modelliert wurde, sorgt dafür, dass der Abgleich robuster durchgeführt werden kann. Ein paarweiser Abgleich aller Eigenschaften jeder Schnittstelle ist manuell extrem aufwendig. Algorithmen könnten zwar Elemente wie den Verbindungstyp oder den Stereotyp des Ports vergleichen, da hier eine Menge an Alternativen vorliegt (Flow, Proxy, Full-Port), jedoch nicht andere Eigenschaften wie den Namen des Ports oder den Freitext. Frei formulierte Angaben wie Port-Name oder Beschreibungen können selbst bei inhaltlicher Kompatibilität unterschiedlich modelliert sein. Semantische Gleichheit kann also vorliegen, obwohl andere Modellelemente oder Freitexte verwendet wurden. Daher reicht ein syntaktischer Abgleich nicht aus, und es muss semantisch überprüft werden, ob die modellierten Elemente die gleiche Bedeutung haben, was durch den Einsatz von Wissensgraphen erreicht werden kann. Ein Wissensgraph besteht aus Entitäten und deren Beziehungen zueinander, die als Knoten und Kanten dargestellt werden /9/. Die Informationen werden als Tripel bestehend aus Subjekt, Prädikat und Objekt im Turtle-Format gespeichert /10/. Die in Tabelle 2 definierten SysML-Elemente können als Tripel gespeichert und im Anschluss mit Hilfe von Large Language Models (LLMs) analysiert werden, um freie Texte und kontextuelle Bedeutungen zu vergleichen. So können die Eigenschaften der Schnittstellen miteinander automatisiert semantisch verglichen werden.

Zusammenfassung und Ausblick

Um die Schnittstellenkompatibilität von Subsystemen zu gewährleisten, müssen unter anderem die Eigenschaften der Schnittstellen miteinander verglichen werden. Dazu wurden auf Grundlage der Literatur zunächst Eigenschaften auf Basis von Verbindungstypen zwischen Subsystemen definiert und somit die erste Forschungsfrage beantwortet. Diese wurden anschließend durch Modellierungsansätzen im MBSE ergänzt, um eine Liste von SysML-Elementen zu erhalten, die für einen Schnittstellenabgleich verwendet werden können, was die zweite Forschungsfrage beantwortet. Der Ansatz ist dadurch limitiert, dass er zwar die zu vergleichenden Elemente definiert, jedoch nicht Aufschluss darüber gibt, wann eine Inkompatibilität vorliegt und wann nicht. Daher soll in den nachfolgenden Forschungsaktivitäten anhand praxisnaher Daten mittels Wissensgraphen diese Überprüfung angewendet werden, um zunächst die Funktionalität und Vollständigkeit der SysML-Elemente zu überprüfen. Anschließend sollen Regeln zur Sicherstellung der Kompatibilität definiert und getestet werden.

Literatur

- /1/ INCOSE: INCOSE systems engineering handbook, John Wiley & Sons, 2023
- /2/ ISO 15288:2023, IEEE, 2023, <https://doi.org/10.1109/IEEESTD.2023.10123367>
- /3/ Shames, P.; Sarrel, M.; Friedenthal, S.: A Representative Application of a Layered Interface Modeling Pattern, INCOSE International Symposium Bd. 26, Nr. 1, Seite 138–159, 2016, <https://doi.org/10.1002/j.2334-5837.2016.00151.x>
- /4/ Bender, B.; Gericke, K.(Hrsg.): Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung. Berlin, Heidelberg : Springer Berlin Heidelberg, 2021, ISBN 978-3-662-57302-0
- /5/ Wilms, R. ; Kronsbein, P. ; Inkermann, D. ; Huth, T. ; Reik, M. ; Vietor, T.: USING A CROSS-DOMAIN PRODUCT MODEL TO SUPPORT ENGINEERING CHANGE MANAGEMENT, Proceedings of the Design Society: DESIGN Conference, S. 1165–1174, 2020 <https://doi.org/10.1017/dsd.2020.90>
- /6/ Madni, A.; Purohit, S.: Economic Analysis of Model-Based Systems Engineering, Systems Bd. 7, Nr. 1, S. 12, 2019, <https://doi.org/10.3390/systems7010012>
- /7/ Fosse, E.; Delp, C. L.: Systems engineering interfaces: A model based approach, IEEE Aerospace Conference, Big Sky MT, IEEE, 2013, <https://doi.org/10.1109/AERO.2013.6497322>
- /8/ Wilms, R.; Cemmasson, V. F.; Inkermann, D.; Reik, M.; Vietor, T.: Identifying Cross-Domain Linkage Types to Support Engineering Change Management and Requirements Engineering, Procedia CIRP Bd. 84, S. 719–724, 2019, <https://doi.org/10.1016/j.procir.2019.04.224>
- /9/ Hogan, A.; Blomqvist, E.; Cochez, M.; D'amato, C.; Melo, G.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; u. a.: Knowledge Graphs, ACM Computing Surveys, Bd. 54, Nr. 4, Seite 1–37, 2022, <https://doi.org/10.1145/3447772>
- /10/ Färber, M.; Bartscherer, F.; Menne, C.; Rettinger, A.: Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, Semantic Web Bd. 9, Nr. 1, Seite 77–129, 2017 <https://doi.org/10.3233/SW-170275>