(19) Deutsches Patent- und Markenamt

(10) **DE 10 2020 215 160 A1** 2022.06.02

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2020 215 160.1**
(22) Anmeldetag: **01.12.2020**
(43) Offenlegungstag: **02.06.2022**

(51) Int Cl.: ***G06F 8/30*** (2018.01)

(71) Anmelder:
Continental Automotive GmbH, 30165 Hannover, DE

(72) Erfinder:
Attenberger, Andreas, 65824 Schwalbach, DE;
Hashem, Mohammed, 65824 Schwalbach, DE;
Sadovan, Iulian, 65824 Schwalbach, DE;
Hernandez Jimenez, Juan Armando, 65824 Schwalbach, DE;
Fiala, Daniel, 65824 Schwalbach, DE

(56) Ermittelter Stand der Technik:

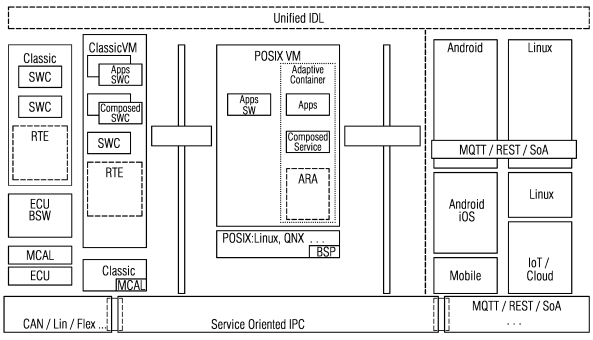
US	**2015 / 0 100 942**	**A1**
US	**2019 / 0 042 215**	**A1**

Rechercheantrag gemäß § 43 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug**

(57) Zusammenfassung: Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug, wobei ein Speichern einer Spezifikationsvorlage für eine Middleware zum Definieren des Typs, des Inhalts und des Verhaltens Middleware in einem computerlesbaren Speichermedium, wobei die Spezifikationsvorlage für jede Middleware als Bausteine eines Metamodells dienen, das alle verfügbaren Bausteine definiert, aus denen ein Zielmodell aufgebaut wird und eine Implementierung der Spezifikationsvorlage erstellt wird und das lauffähige Computerprogramm der Implementierung automatisch erzeugt wird.



Beschreibung

FELD

[0001] Die vorliegende Erfindung betrifft ein Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug

HINTERGRUND

[0002] Die Erfindung betrifft die Bereiche Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug und Softwareentwicklung. Insbesondere handelt es sich um eine Vorrichtung und ein Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug.

[0003] Ziel der Erfindung ist es, das Schreiben von Computercode, insbesondere in der Automotive-Welt zu verbessern, und das Implementieren von Anwendungsprogrammen zu vereinfachen und zu beschleunigen. Um dies zu erreichen, ist ein bestimmter Satz von Grundelementen erforderlich, der ein Spezifikationsmuster implementiert, mit dem die Arten von Fahrzeuganwendungsprogrammen für vollständig definiert werden können. Was benötigt wird, ist eine Reihe von Grundelementen und ein Editor-Tool, mit dem die Benutzeroberfläche mithilfe eines Modells angegeben, mit dem Benutzer validiert, auf Vollständigkeit und Nichtmehrdeutigkeit überprüft und automatisch in Quellcode in einer Programmiersprache der dritten Generation umgewandelt werden kann Arbeitscomputercode auszugeben, der eine solche Benutzeroberfläche implementiert, die in der Lage ist, mit einer Geschäftslogikkomponente zu kommunizieren, die für die Implementierung der Funktionalität der Anwendung verantwortlich ist, für die die Benutzeroberfläche das „Front-End“ ist.

[0004] Die Probleme mit Spezifikationen und Tools zur Codegenerierung nach dem Stand der Technik sind folgende:

> Spezifikationsmethoden berücksichtigen das Domänenmodell, d.h. die Funktionalität der jeweiligen Anwendung. Beispiele für solche dann fehlerhaften Modell- und Codegenerierungswerkzeuge sind bekannt. Mit anderen Worten, diese Modellierungswerkzeuge des Standes der Technik setzen voraus, dass die Implementierungslogik vollständig spezifiziert und manuell codiert wird.

[0005] Das separate Codieren der Implementierungslogik kann zu Fehlern bei der Integration und zu Fehlern im Fahrzeugfunktionscode selbst führen. Dies verlängert die Verzögerung von der Konzeption bis zum ersten Verkauf auf dem Markt (Time-to-Market), da möglicherweise lange Verzögerungen beim Debuggen des Fahrzeugfunktionalitätscodes auftreten. Darüber hinaus verfügen diese Spezifikationstools nach dem Stand der Technik nicht über eine präzise, eindeutige Semantik, die validiert werden kann, um Fehler im Code zu beseitigen, Code automatisch aus der validierten Spezifikation generieren und mit einer generierten Verbindung verbinden setzen zu können.

[0006] Andererseits berücksichtigen Methoden und Notationen zur Spezifikation der funktionalen Anforderungen von Computerprogrammen, wie Tools zum Definieren von Modellen der Logik eines Programms, wie OMT oder UML, auch die Anforderungen für Fahrzeugfunktionscodes nicht.

[0007] Diese Situation führt zu einem Zustand, in dem mindestens zwei Modelle benötigt werden, um die Geschäftslogik der Modelle und das Fahrzeugfunktionscodes eines Computerprogramms vollständig zu spezifizieren. Dies führt zu Synchronisationsproblemen, wenn sich eine der Spezifikationen ändert. Mit anderen Worten, wenn sich eine der Spezifikationen ändert, „bricht“ die andere Spezifikation und der geschriebene Code, der diese andere Spezifikation implementiert, funktioniert nicht mehr mit dem Code, der geschrieben wurde, um die erste Spezifikation zu implementieren. Dies erfordert eine teure und zeitaufwendige Überarbeitung der zweiten Spezifikation.

[0008] Generierungswerkzeuge erzeugen Teilcode, bei dem der generierte UI-Code nicht vollständig ist. Ein solcher Code kann nicht direkt kompiliert werden und muss von Hand codiert werden, bevor ein Programm existiert, das als endgültige Anwendung ausgeführt werden kann.

[0009] Modellierungswerkzeuge enthalten keine Grundelemente, die aufgerufen werden können, um die Elemente eines Fahrzeugfunktionscodes anzugeben.

[0010] Die Werkzeuge des Standes der Technik schlagen jedoch Grundelemente auf sehr niedrigem Niveau vor, die nicht in mittelgroßen Industrieprojekten verwendet werden können, ohne Skalierbarkeitsprobleme zu haben. Akademische Forschung zu Modellierungswerkzeugen wie iUML haben zu Werkzeugen geführt, die nicht in Industrieprojekten getestet und nicht auf solche Skalierbarkeitsanforderungen abgestimmt wurden.

[0011] Einige Notationen wie OVID und UML haben in den verwendeten Konzepten keine genaue semantische Bedeutung.

[0012] Infolgedessen ist es unmöglich, validierte Spezifikationen des und fehlerfreien Codes aus einer ungenauen Spezifikation zu generieren.

[0013] US 5185867 (A) beschreibt ein Informationsverarbeitungssystem zum Erzeugen von Softwarespezifikationen, einem Verfahren und einer Vorrichtung zum automatischen Erzeugen von Softwarespezifikationen, zum Erleichtern von Softwarewartungsarbeiten durch Erzeugen einer Spezifikation aus mindestens einem Softwareprodukt, das durch Spezifizieren der Spezifikation in der Softwareentwicklung erhalten wird. Das Verfahren umfasst die Schritte des Lesens von Informationen, die in mindestens einem Softwareprodukt beschrieben sind, das durch Spezifizieren einer Spezifikation mit hohem Rang erhalten wurde, Extrahieren von Informationen, die von der Spezifikation mit hohem Rang übernommen wurden, zu Spezifikationen von niedrigem Rang aus den so gelesenen Informationen, Teilen der extrahierten Informationen in gemeinsame Informationen Elemente und Elemente, die einzeln vorhanden sind und sich auf die gemeinsamen Elemente beziehen, einzeln vorhandene Elemente für jedes gemeinsame Element zusammenstellen, entsprechende Elemente auf der Grundlage der Informationen anordnen, die für jedes gemeinsame Element in einer Netzwerkform zusammengestellt wurden, Netzwerkinformationen generieren, Darstellungsformat von konvertieren jedes Element der Netzwerkinformationen und Generieren einer hochrangigen Spezifikation.

[0014] Im Fahrzeug und den angrenzenden Bereichen zum Beispiel Cloud, Mobile Apps, setzen unterschiedliche Softwaretechnologien und Kommunikations-Middelwares. Solche Kommunikations-Middlewares sind z.B. AUTOSAR, Genivi, AGL, Android AUTO. Für jede dieser Domänen sind somit auch unterschiedliche Schnittstellenbeschreibungen, wie AUTOSAR xml, Franca, Protobuff nötig, wie es in **Fig. 1** angedeutet ist. In neuen Fahrzeuggenerationen werden einzelne Funktionalitäten auch verschiedenen Steuergeräten und somit auch verschiedenen Kommunikations-Middlewares verteilt. Stand heute gibt es keine Möglichkeit eine Schnittstellenbeschreibung für die gesamte Fahrzeugbandbreite zu erstellen, welche ein direktes Ableiten der spezifischen Formate der einzelnen eingesetzten Technologien erlaubt.

BESCHREIBUNG UND VORTEILE

[0015] Die technische Aufgabenstellung ergibt sich aus der Kenntnis der Mängel. Die derzeit vorhandenen Tools zur Software Schnittstellendefinition beschränken sich entweder auf spezifische Bereiche, wie bei AUTOSAR (In-Car) oder haben kein Link zur eigentlichen Implementierung, welches zum Beispiel bei Standard UML Tools der Fall ist.

[0016] Durch die fehlende Darstellung der gesamten Fahrzeugschnittstellen in einem einheitlichen Format ist die Verifikation gegen ein Designmodell sehr schwierig Es gibt derzeit keine Möglichkeiten das Routen von einer Kommunikationsart zu einer anderen einheitlich darzustellen, (Signal ≠ Service/-Service ≠ Pub/Sub/Service ≠ Cloud).

[0017] Das Herauslösen der einzelnen Kommunikations-Middlewarelösungen wäre Lösungen aus ihren „Silos“, um eine Möglichkeit der einheitlichen Beschreibung am Aufbau inklusive Routen zu ermöglichen.

[0018] Hierdurch entstehen Kompatibilitätsprobleme der einzelnen Software Schnittstellen durch verschiedene Beschreibungsformate und es existiert kein durchgängiges Tool-Workflow von Design bis zur Verifikation.

[0019] Aufgabe der Erfindung ist es eine effiziente Softwareentwicklung zu ermöglichen, die den gegenwärtigen Einschränkungen durch die Domänenentwicklung überwindet.

[0020] Die Aufgabe wird gelöst durch das Verfahren nach Anspruch 1

[0021] Vorteilhaft wird die Aufgabe gelöst durch ein Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug, wobei ein Speichern einer Spezifikationsvorlage für eine Middleware zum Definieren des Typs, des Inhalts und des Verhaltens Middleware in einem computerlesbaren Speichermedium, wobei die Spezifikationsvorlage für jede Middleware als Bausteine eines Metamodells dienen, das alle verfügbaren Bausteine definiert, aus denen ein Zielmodell aufgebaut wird, eine Implementierung der Spezifikationsvorlage erstellt wird und das lauffähige Computerprogramm der Implementierung automatisch erzeugt wird, ein Laden von Daten aus einer Datenstruktur und die eine Spezifikation in einer formalen Sprache umfasst, die die Muster definiert, die den Typ, den Inhalt und das Verhalten aller Elemente der gewünschten Benutzerschnittstelle für das Computerprogramm definieren, wobei das Laden von Daten aus computerlesbares Speichermedium erfolgt, um jede Spezifikationsvorlage gemäß der tatsächlich gewünschten Anwendung der der Spezifikationsvorlage anzupassen, ein Laden von Daten aus einer Software Beschreibungsumgebung die die Anwendung beschreibt, Durchführen einer Transformation an diesen angepassten Vorlagen, um sicherzustellen, dass fehlende Informationen geliefert werden, und mit den Daten aus der einer Software Beschreibungsumgebung verglichen werde, Durchführen einiger vorbestimmter vorläufiger Berechnungen, in der die Daten überprüft werde, Untersuchen der angepassten Vorlagen in einem Speichermedium, um zu bestimmen, welche von mehreren Vorlagen in einer endgültigen Zielprogrammiersprache auf hoher Ebene verwendet werden kann, um jede

der angepassten Vorlagen zu implementieren, und Bestimmen der Vorlage in einer endgültigen Ausgabedatei eines Computers auf hoher Ebene Code in der Ziel-Computerprogrammiersprache, die der zentrale Serverarchitektur ausführbar ist, durch Extrahieren der erforderlichen Informationen aus der angepassten Vorlage, um für mindestens eine Spezifikationsvorlage den Computercode der formalen Spezifikation, der die Spezifikationsvorlage implementiert, zu erzeugen.

[0022] Zur Beschreibung der Software Schnittstellen wird bereits vorhandene Interface Description Language (IDL) in einem Verfahren zu einer sogenannten“ Unified IDL" zu herangezogen und diese Unified IDL bildet eine Beschreibungsschicht über den tatsächlich eingesetzten Middlewares.

TECHNISCHE VORTEILE DER ERFINDUNG

[0023] Mithilfe der Unified IDL lassen sich alle relevanten Softwareschnittstellen middlewareunabhängig beschreiben. Mithilfe von eingeführten Transformatoren lassen sich die aus der Unified IDL Beschreibungen, die in den Zielsystemen eingesetzten Beschreibungsformate generieren.

[0024] Erstmals wird eine Unified IDL über den gesamte Fahrzeugbandbreite angewendet vom Sensor bis zur Cloud und gegen das Unified IDL Modell kann die spätere Implementierung mit einem geeigneten Tool getestet werden.

[0025] Vorteilhaft erfolgt hierdurch eine Verkürzung der Entwicklungszeiten durch vereinfachte Migration und Wiederverwendbarkeit von Software, sowie die vereinfachte Kommunikation zwischen den Fahrzeugdomänen und Firmen durch einheitliches Austauschformat. Weitergehend wird eine vereinfachte Freigabe/Homologation der Fahrzeugsoftware durch einheitliche Darstellung der Software Applikationsarchitektur und einfache Möglichkeit der Verifikation der Softwarearchitektur gegen ein Modell, welches ECU-übergreifend ist ermöglicht. Desweitern erfolgt vorteilhaft ein vereinfachter Austausch von Software Applikationen zwischen Firmen und den an der Software Entwicklung Beteiligten durch einheitliche Schnittstellenbeschreibung.

[0026] In einer vorteilhaften Ausgestaltung zeichnet sich das Verfahren dadurch gekennzeichnet, dass die Daten aus einer Software-Beschreibungsumgebung aus einem UML Tools sind.

[0027] In einer weiteren vorteilhaften Ausgestaltung zeichnet sich das Verfahren dadurch aus, dass die Transformation bidirektional erfolgen kann.

[0028] In einer besonders vorteilhaften Ausgestaltung erfolgt die Transformation bidirektional.

[0029] Eine weitere besonders vorteilhaften Ausgestaltung des erfindungsgemäßen Verfahrens zeichnet sich dadurch aus, dass die Daten aus der Software-Beschreibungsumgebung mit der Ausgabedatei des Computers auf hoher Ebene Code in der Ziel-Computerprogrammiersprache, die auf der zentrale Serverarchitektur ausführbar ist regelungstechnisch gekoppelt sind.

Figurenliste

[0030] Ein Ausführungsbeispiel der Erfindung ist in den Zeichnungen dargestellt und wird im Folgenden näher beschrieben. Es zeigen:

Fig. 1 Steuergeräte und Sensoren in einem gegenwärtigen Fahrzeugaufbau,

Fig. 2 Integration der Rechner Funktionalität in einer ECU,

Fig. 3 Darstellung des Integrationsschritt zur Zonen ECU,

Fig. 4 gesamte Integration in einem zentralen High Performance Server,

Fig. 5 gegenseitiger systematischer Aufbau in einem Ablauf zur Softwaregenerierung,

Fig. 6 Darstellung der Barrieren in der gegenwärtigen Systematik,

Fig. 7 schematische Darstellung der möglichen Verarbeitung mittels des erfindungsgemäßen Ansatzes,

Fig. 8 schematische Darstellung der erfindungsgemäßen Überlegung,

Fig. 9 schematische Darstellung des erfindungsgemäßen Verfahrens,

Fig. 10 Systemdarstellung der Erfindung,

Fig. 11 Darstellung der Integration anhand der zu implementierenden Prozesse.

BESCHREIBUNG VON AUSFÜHRUNGSBEISPIELEN

[0031] In der **Fig. 1** bis ist schematisch gedeutet, wie sich die Entwicklung von einer aus Steuergerät und Sensor Funktionsarchitektur über den Domänenansatz zu der High Performance Server Architektur gelangt. **Fig. 5** beschreibt in einer Blockdiagrammstruktur die Gesamtsituation, wie in der gegenwärtigen Softwareentwicklung zur Implementierung von Fahrzeugfunktionssoftware die einzelnen Komponenten zu interagieren haben und welche Schnittstellen zu berücksichtigen sind. Wie in der **Fig. 5** angegeben bezeichnet, sind die geläufigen Abkürzungen, die der Fachmann als Terminologie kennt.

[0032] Ein gegebenes Problem kann auf viele Arten implementiert werden, indem unterschiedliche Analyse-, Entwurfs- und Implementierungsentscheidungen getroffen werden.

[0033] Die Abstraktionstrennung zwischen der Analyse und der Implementierung wird als Problemraum bzw. Lösungsraum bezeichnet.

[0034] In diesem Szenario ist das konzeptionelle Modell ein Satz von Objekten, die das gewünschte Verhalten verschiedener Aspekte des Codes definieren, wie beispielsweise ein Objektmodell, ein dynamisches Modell, ein Funktionsmodell, die genommen wurden, um Sie gemeinsam die Anforderungen des zu schreibenden Programms oder Systems anpassen.

[0035] Eine Spezifikation in einer formalen Sprache wird unter Verwendung des konzeptionellen Modells erstellt und kann nach der Validierung gemäß den in einem Ausführungsmodell beschriebenen Regeln in eine Implementierung (Arbeitscode) übersetzt werden.

[0036] Das Objektmodell, das dynamische Modell und das Funktionsmodell werden verwendet, um die strukturellen und funktionalen Anforderungen des zu schreibenden Programms oder Systems zu beschreiben.

[0037] Ein mögliche Benutzeroberflächenmodell kann eingesetzt, um die Spezifikation durch Benutzeroberflächenanforderungen zu ergänzen, die den Mechanismus definieren, durch den der Benutzer Informationen von dem zu erstellenden Programm erhält und ihm Informationen bereitstellt.

[0038] Die Modelle im konzeptionellen Modell 104 werden von Programmieren (im Folgenden als Datenanalysten bezeichnet) unter Verwendung eines Editors (auch als Modeler bezeichnet) erstellt, der speziell für die Konstruktion, dargestellt durch Pfeil UML Tool in **Fig. 10** der Modelle im konzeptionellen Modell erstellt wurde .

[0039] Mit anderen Worten, der Editor verfügt über Werkzeuge, Symbole, Menüoptionen, Dialogfelder usw., mit denen die verschiedenen Bausteine in einem Metamodell, aus denen alle konzeptuellen Modelle erstellt werden, ausgewählt und in das konzeptionelle Modell eingefügt werden können. Der Editor ermöglicht auch die Definition der Attribute der Objekte im Konzeptmodell und die Definition der Beziehungen zwischen Objekten im Konzeptmodell.

[0040] Objekte wie dieser Begriff werden hier verwendet und bedeuten Objekte im Sinne der objektorientierten Computerprogrammierung.

[0041] Jedes Objekt verfügt über eine Datenstruktur, Prozeduren oder Programme, die an der Datenstruktur arbeiten, und einen Mechanismus zum Übersetzen von Nachrichten.

[0042] Sobald das konzeptionelle Modell erstellt ist, wird es in eine Spezifikation in einer formalen Sprache (im Folgenden als formale Spezifikation oder nur als Spezifikation bezeichnet) wie OASIS übersetzt, es kann jedoch jede formale Sprache verwendet werden.

[0043] Die Verwendung einer formalen Sprache ist wichtig, da sie Mehrdeutigkeiten beseitigt, da es Grammatikregeln (Syntax plus Semantik) gibt, mit denen die Spezifikation validiert werden kann, um sicherzustellen, dass sie vollständig und korrekt ist.

[0044] Die formalen Spezifikationen können in einem Speicher (Computerspeicher oder Festplatte) gespeichert werden, das eine relationale Datenbank, eine Binärdatei oder eine andere binäre Darstellung der Informationen sein kann.

[0045] Die bevorzugte Ausführungsform ist eine XML-Form.

[0046] Die formale Spezifikation wird von einem Validator validiert, um Vollständigkeit, Korrektheit und Nichtmehrdeutigkeit zu überprüfen, und dieser Validierungsprozess ist der Code, der fehlerfrei ist, wenn er automatisch von einem Übersetzer erstellt wird.

[0047] Dies wäre nicht möglich, wenn die Spezifikation nicht in einer formalen Sprache mit strengen, nicht variablen Regeln für Syntax und Semantik geschrieben wäre.

[0048] Durch die Verwendung dieser Grammatikregeln der formalen Sprache kann die Erklärung in der formalen Spezifikation überprüft werden, um sicherzustellen, dass jede Aussage vollständig, korrekt und nicht mehrdeutig ist.

[0049] Sobald klar ist, dass jede Aussage in der formalen Spezifikation vollständig, korrekt und nicht mehrdeutig ist, ist der resultierende Code fehlerfrei.

[0050] Mit anderen Worten, wenn die Spezifikation gültig ist, kann sie als Eingabe für Codegeneratoren oder automatische Übersetzer verwendet werden, die das Ausführungsmodell implementieren, um Anwendungen für eine von mehreren Sprachen und / oder Plattformen, Visual Basic / Windows, Java Swing / Java MV, ColdFusion MX / Web-Apps, JSP / Web-Apps zu erzeugen.

[0051] Fig. 11 wird anhand von verschiedenen Funktionsblöcken darstellt, wie die - Modellierung

der Kommunikation von Softwareanwendungen über heterogene Systeme, Netzwerke und Paradigmen hinweg erfolgt und eine Entwicklung von kommunikationsmiddleware-unabhängigen Anwendungen realisiert wird. Die Migration von Schnittstellenbeschreibungen zwischen automobilen Kommunikations-Middleware's erfolgt als Hauptfunktion und hierdurch erlangt man eine Erleichterung des Informationsaustausches von Kommunikationsarchitekturen zwischen der an der Softwareentwicklung beteiligten Parteien.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Zitierte Patentliteratur

- US 5185867 A [0013]

Patentansprüche

1. Verfahren zum Erzeugen von Computercode für eine zentrale Serverarchitektur in einem Fahrzeug, dadurch kennzeichnet, dass
- Speichern einer Spezifikationsvorlage für eine Middleware zum Definieren des Typs, des Inhalts und des Verhaltens Middleware in einem computerlesbaren Speichermedium, wobei die Spezifikationsvorlage für jede Middleware als Bausteine eines Metamodells dienen, das alle verfügbaren Bausteine definiert, aus denen ein Zielmodell aufgebaut wird,
- eine Implementierung der Spezifikationsvorlage erstellt wird und das lauffähige Computerprogramm der Implementierung automatisch erzeugt wird,
- Laden von Daten aus einer Datenstruktur und die eine Spezifikation in einer formalen Sprache umfasst, die die Muster definiert, die den Typ, den Inhalt und das Verhalten aller Elemente der gewünschten Benutzerschnittstelle für das Computerprogramm definieren, wobei das Laden von Daten aus computerlesbares Speichermedium erfolgt, um jede Spezifikationsvorlage gemäß der tatsächlich gewünschten Anwendung der der Spezifikationsvorlage anzupassen,
- Laden von Daten aus einer Software-Beschreibungsumgebung, die die Anwendung beschreibt,
- Durchführen einer Transformation (T) an diesen angepassten Vorlagen, um sicherzustellen, dass fehlende Informationen geliefert werden, und mit den Daten aus der einer Software-Beschreibungsumgebung verglichen werde,
- Durchführen einiger vorbestimmter vorläufiger Berechnungen, in der die Daten überprüft werde,
- Untersuchen der angepassten Vorlagen in einem Speichermedium, um zu bestimmen, welche von mehreren Vorlagen in einer endgültigen Zielprogrammiersprache auf hoher Ebene verwendet werden kann, um jede der angepassten Vorlagen zu implementieren, und
- Bestimmen der Vorlage in einer endgültigen Ausgabedatei eines Computers auf hoher Ebene Code in der Ziel-Computerprogrammiersprache, die der zentrale Serverarchitektur ausführbar ist, durch Extrahieren der erforderlichen Informationen aus der angepassten Vorlage, um für mindestens eine Spezifikationsvorlage den Computercode der formalen Spezifikation, der die Spezifikationsvorlage implementiert, zu erzeugen.

2. Verfahren nach Anspruch 1, dadurch kennzeichnet, dass die Daten aus einer Software-Beschreibungsumgebung aus einem UML Tools sind.

3. Verfahren nach Anspruch 1 oder 2, dadurch kennzeichnet, dass die Transformation bidirektional erfolgen kann.

4. Verfahren nach einem der vorgehenden Ansprüche 1 bis 3, dadurch kennzeichnet, dass die Transformation bidirektional erfolgen kann.

5. Verfahren nach einem der vorgehenden Ansprüche 1 bis 4, dadurch kennzeichnet, dass die der Daten aus der Software-Beschreibungsumgebung mit der Ausgabedatei des Computers auf hoher Ebene Code in der Ziel-Computerprogrammiersprache, die auf der zentrale Serverarchitektur ausführbar ist regelungstechnisch gekoppelt sind.

Es folgen 12 Seiten Zeichnungen

Anhängende Zeichnungen

FIG 1

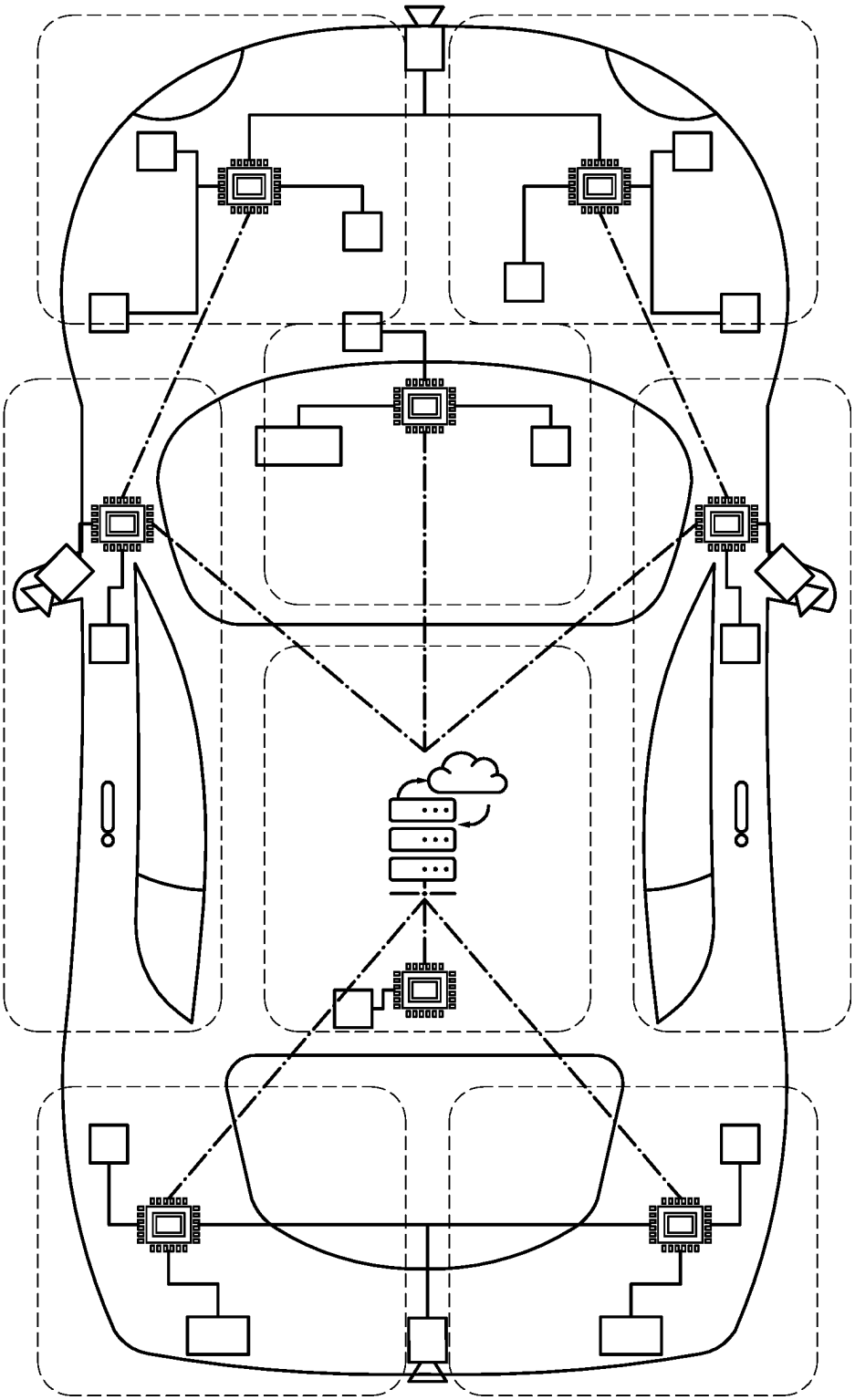


FIG 2

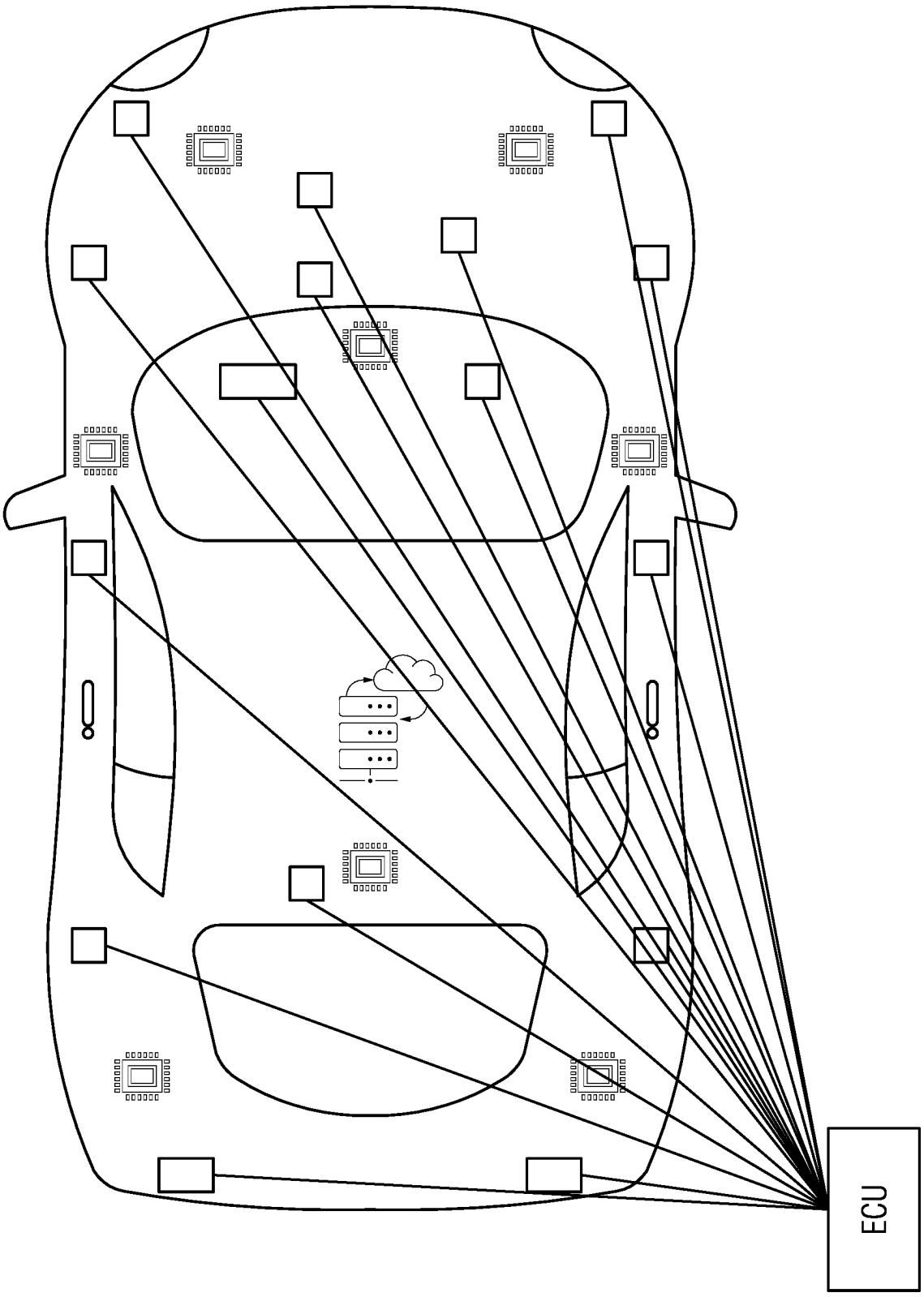


FIG 3

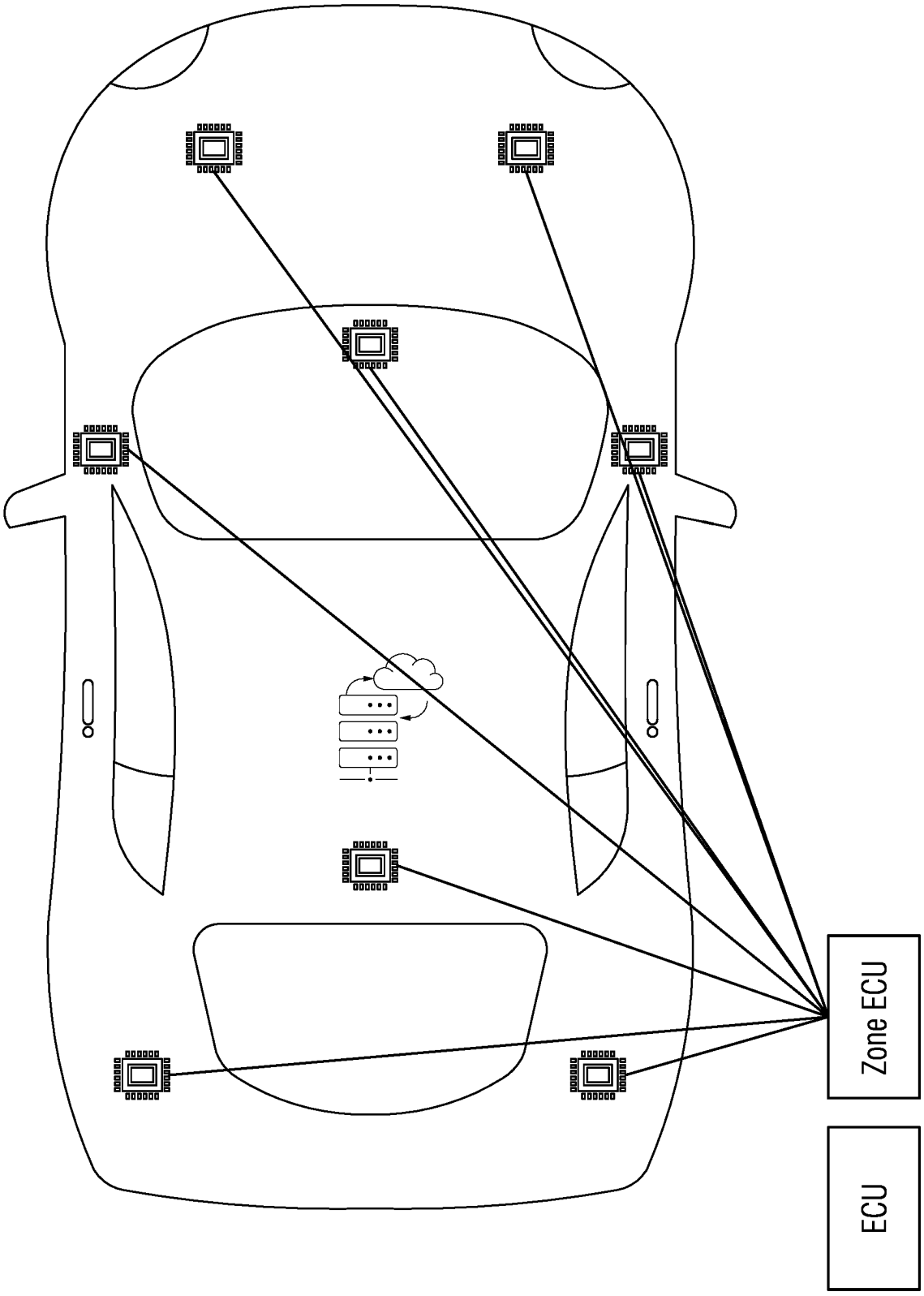


FIG 4

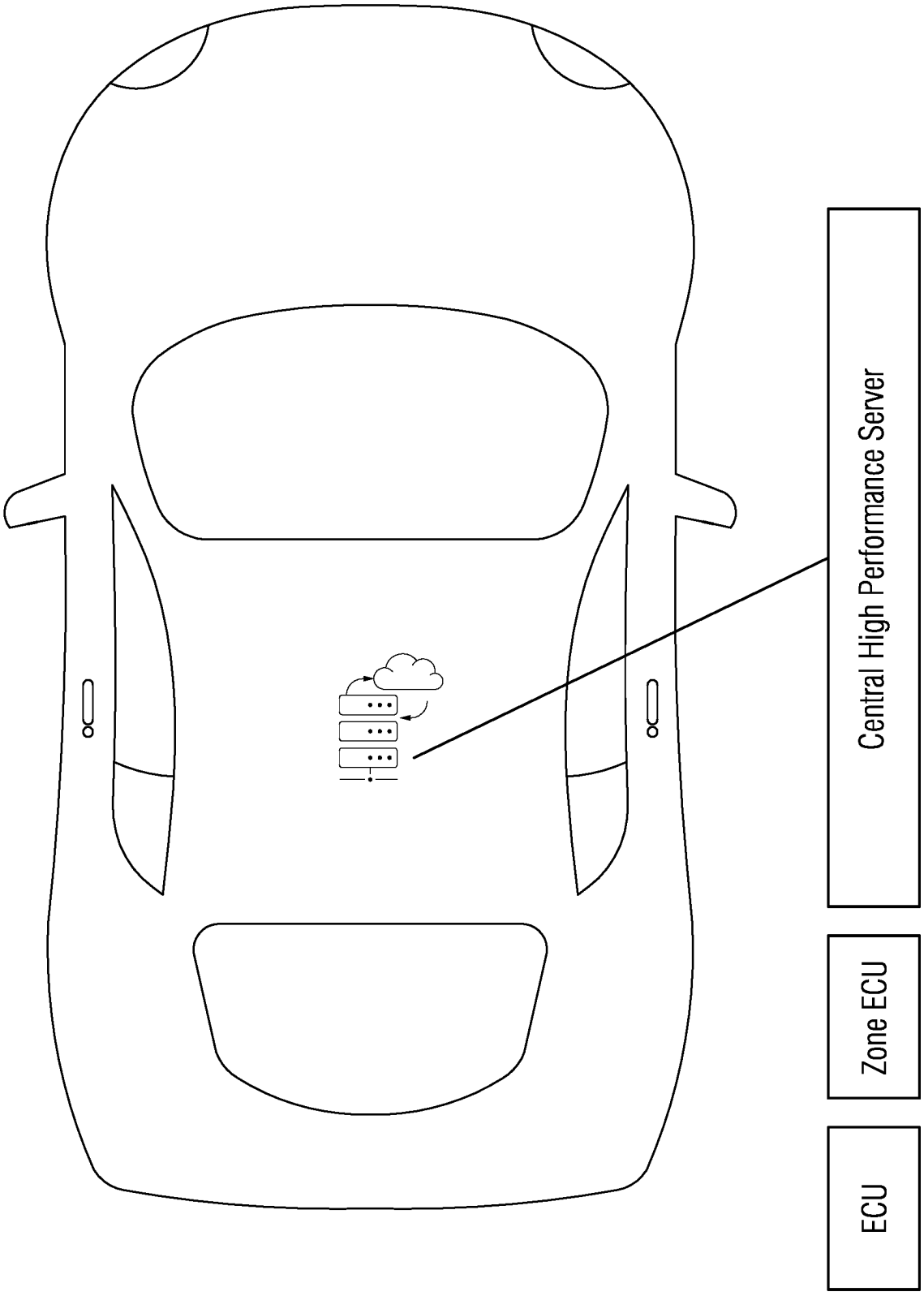
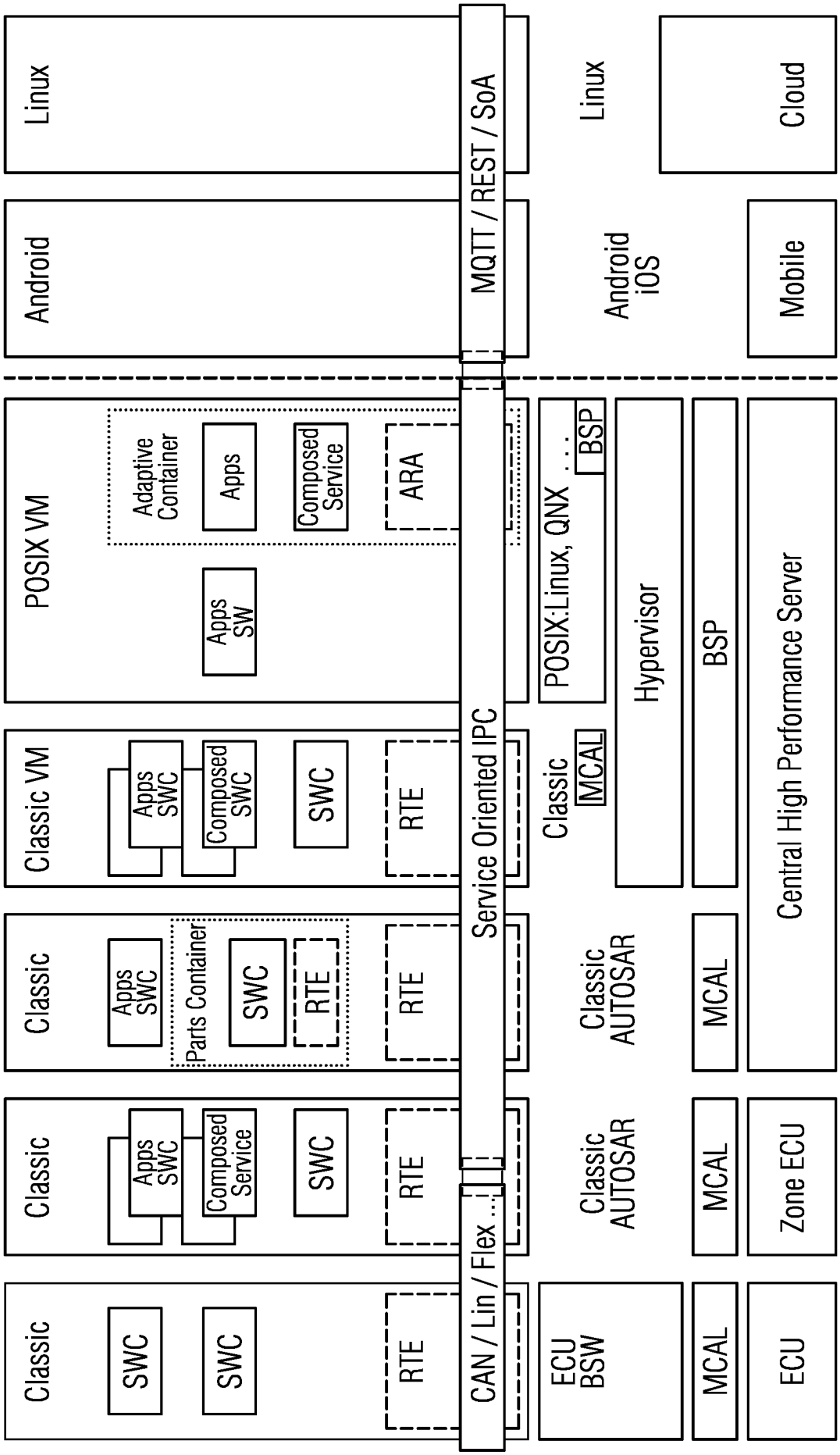


FIG 5


Classic
SWC
SWC
RTE
ECU
BSW
MCAL
ECU
Classic
Apps SWC
Composed Service
SWC
RTE
CAN / Lin / Flex ...
Classic AUTOSAR
MCAL
Zone ECU
Classic
Apps SWC
Parts Container
SWC
RTE
RTE
Classic AUTOSAR
MCAL
Central High Performance Server
Classic VM
Apps SWC
Composed SWC
SWC
RTE
Service Oriented IPC
Classic
MCAL
POSIX:Linux, QNX ...
BSP
Hypervisor
BSP
POSIX VM
Adaptive Container
Apps
Composed Service
ARA
Apps SW
Android
MQTT / REST / SoA
Android iOS
Mobile
Linux
Linux
Cloud

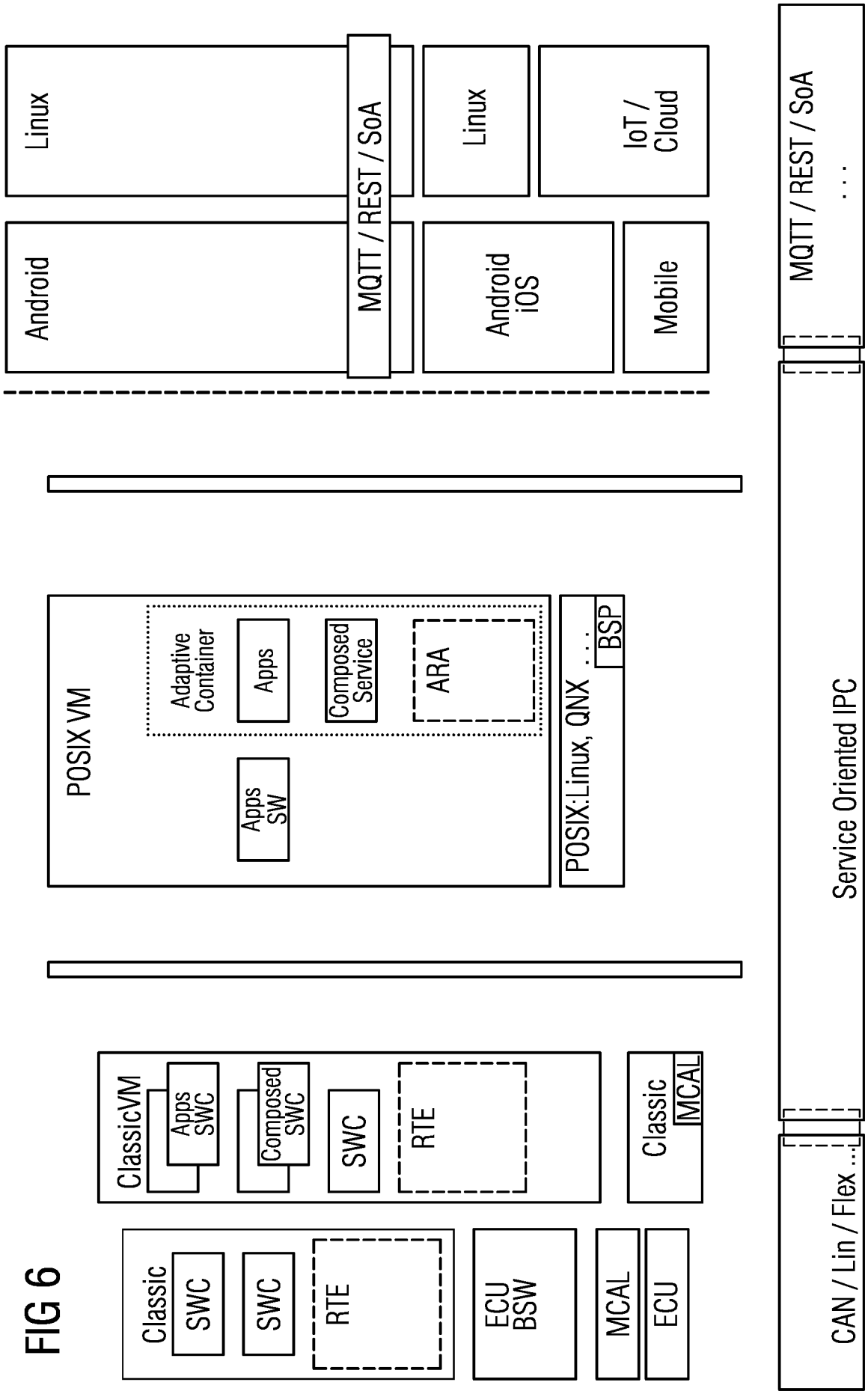

Linux
Android
MQTT / REST / SoA
Linux
Android iOS
IoT / Cloud
Mobile
MQTT / REST / SoA
. . .
POSIX VM
Adaptive Container
Apps
Composed Service
ARA
Apps SW
POSIX:Linux, QNX . . .
BSP
Service Oriented IPC
ClassicVM
Apps SWC
Composed SWC
SWC
RTE
Classic
MCAL
Classic
SWC
SWC
RTE
ECU BSW
MCAL
ECU
CAN / Lin / Flex ...

FIG 6

FIG 7

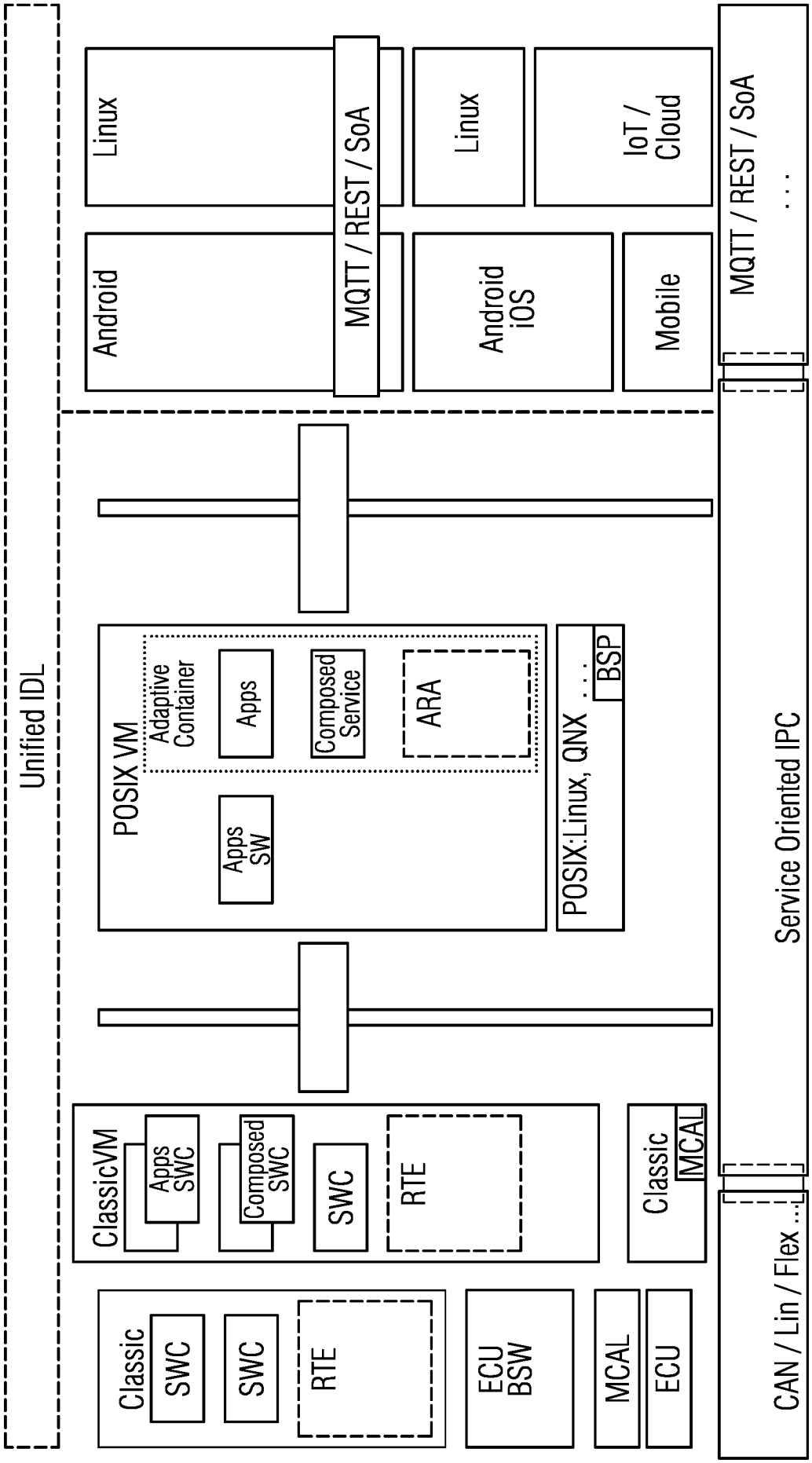

Unified IDL
Linux
Android
MQTT / REST / SoA
Linux
Android iOS
IoT / Cloud
Mobile
MQTT / REST / SoA
. . .
POSIX VM
Adaptive Container
Apps
Composed Service
ARA
Apps SW
POSIX:Linux, QNX . . .
BSP
Service Oriented IPC
ClassicVM
Apps SWC
Composed SWC
SWC
RTE
Classic
MCAL
Classic
SWC
SWC
RTE
ECU BSW
MCAL
ECU
CAN / Lin / Flex ...

FIG 8

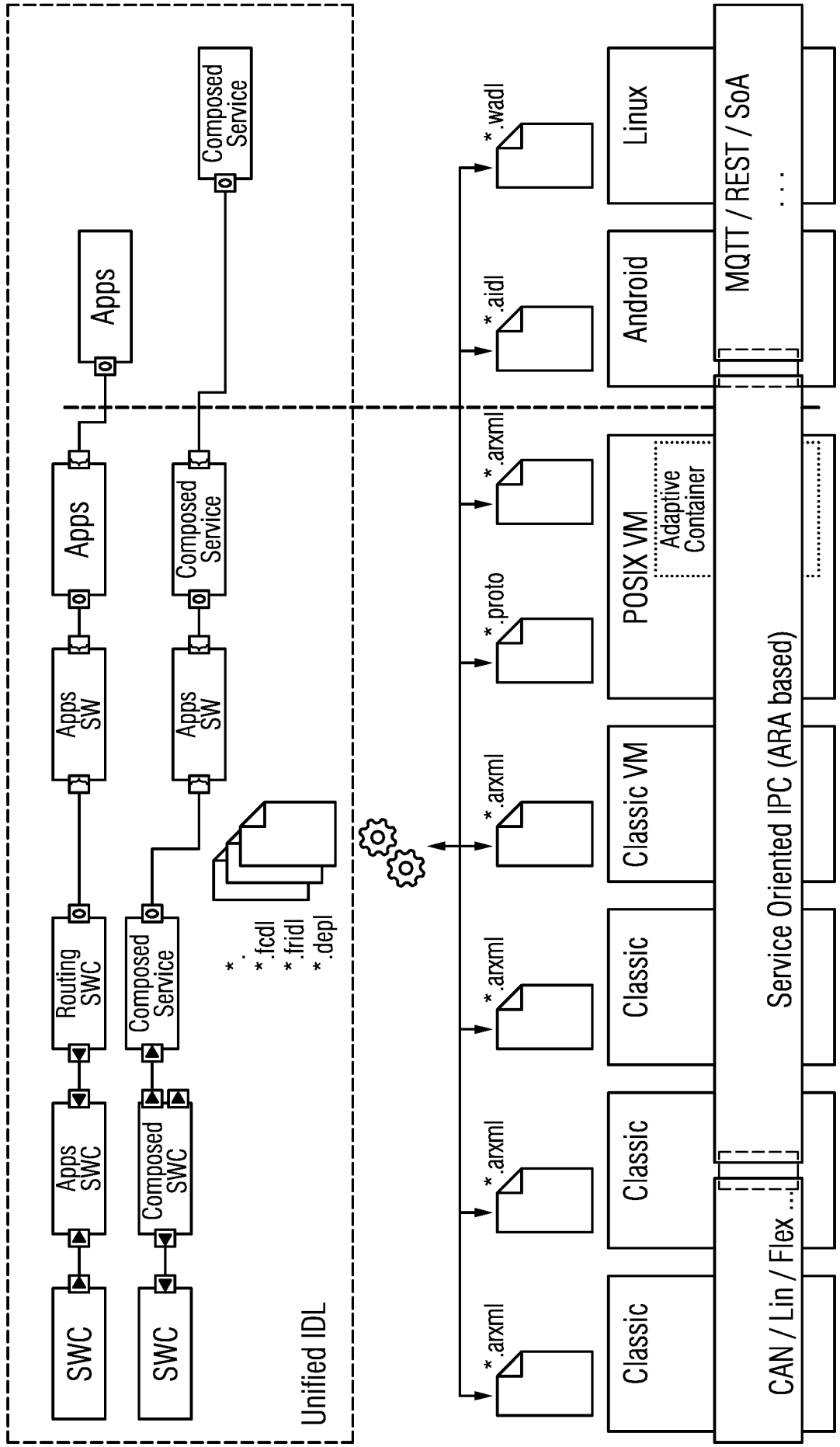
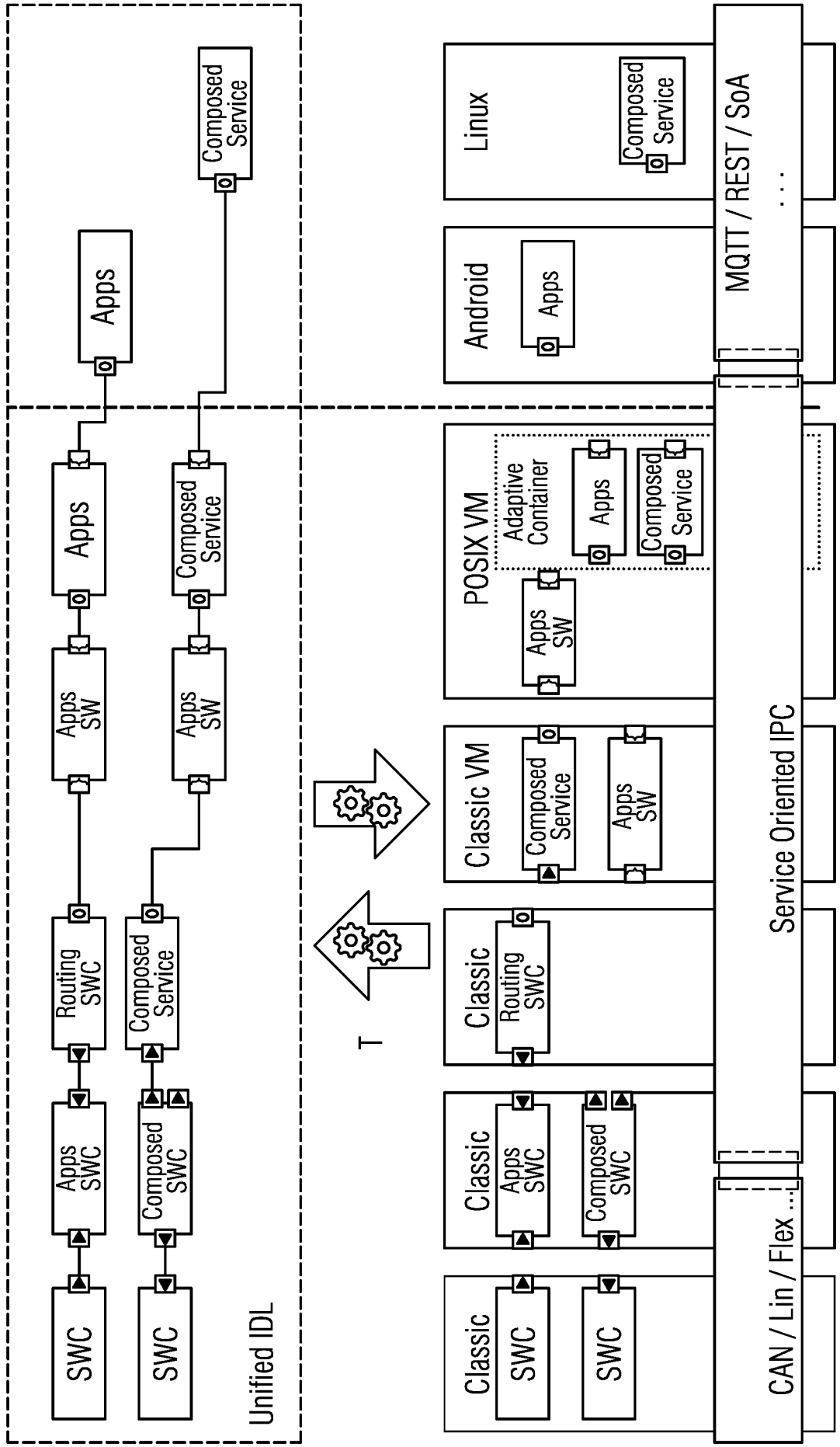

Unified IDL
SWC
SWC
Apps SWC
Composed SWC
Routing SWC
Composed Service
Apps SW
Apps SW
Apps
Apps
Composed Service
Composed Service
*.fcdl
*.fridl
*.depl
*.arxml
*.arxml
*.arxml
*.arxml
*.arxml
*.proto
*.aidl
*.wadl
Classic
Classic
Classic
Classic VM
POSIX VM
Adaptive Container
Android
Linux
CAN / Lin / Flex ...
Service Oriented IPC (ARA based)
MQTT / REST / SoA
. . .

FIG 9


Unified IDL
SWC
SWC
Apps SWC
Composed SWC
Routing SWC
Composed Service
Apps SW
Apps SW
Apps
Composed Service
Apps
Composed Service
T
Classic
SWC
SWC
Classic
Apps SWC
Composed SWC
Classic
Routing SWC
Classic VM
Composed Service
Apps SW
POSIX VM
Apps SW
Adaptive Container
Apps
Composed Service
Android
Apps
Linux
Composed Service
CAN / Lin / Flex ...
Service Oriented IPC
MQTT / REST / SoA
. . .

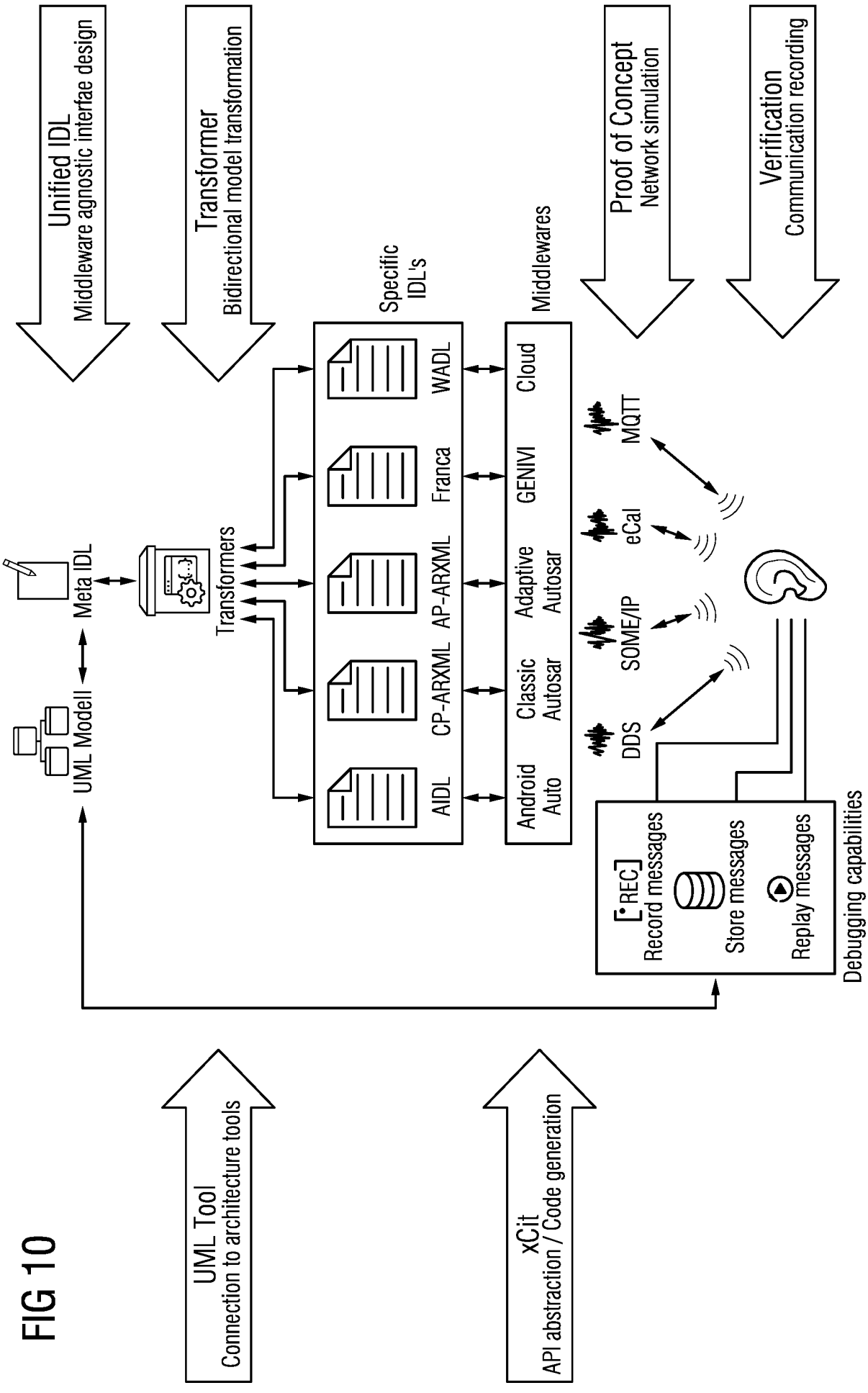
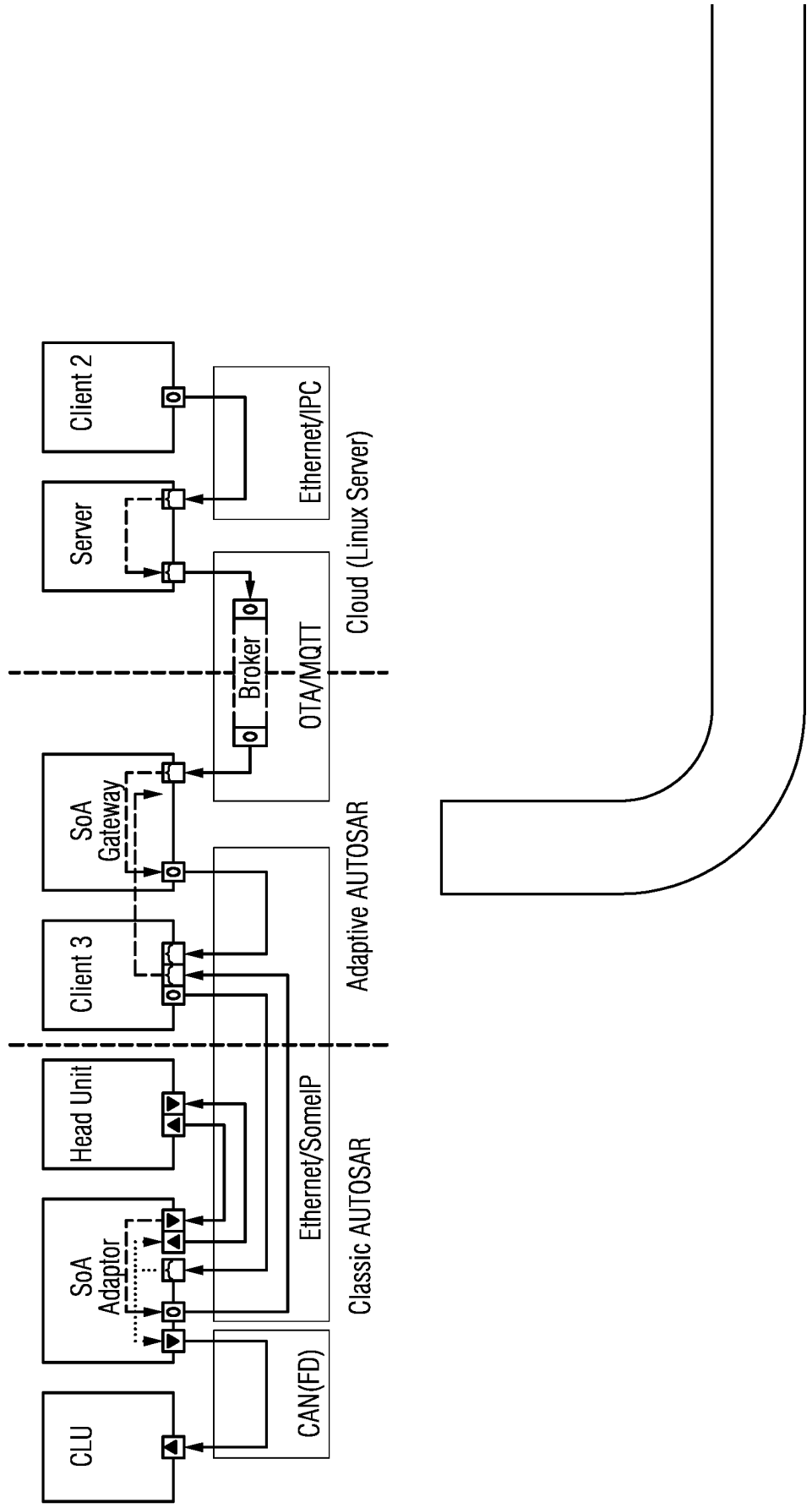

Unified IDL
Middleware agnostic interfae design
Transformer
Bidirectional model transformation
Proof of Concept
Network simulation
Verification
Communication recording
Specific
IDL's
Middlewares
Meta IDL
UML Modell
Transformers
AIDL
CP-ARXML
AP-ARXML
Franca
WADL
Android
Auto
Classic
Autosar
Adaptive
Autosar
GENIVI
Cloud
DDS
SOME/IP
eCal
MQTT
[REC]
Record messages
Store messages
Replay messages
Debugging capabilities
UML Tool
Connection to architecture tools
xCit
API abstraction / Code generation

FIG 10

FIG 11 - Part 1


CLU
SoA Adaptor
Head Unit
Client 3
SoA Gateway
Server
Client 2
CAN(FD)
Ethernet/SomeIP
Broker
OTA/MQTT
Ethernet/IPC
Classic AUTOSAR
Adaptive AUTOSAR
Cloud (Linux Server)

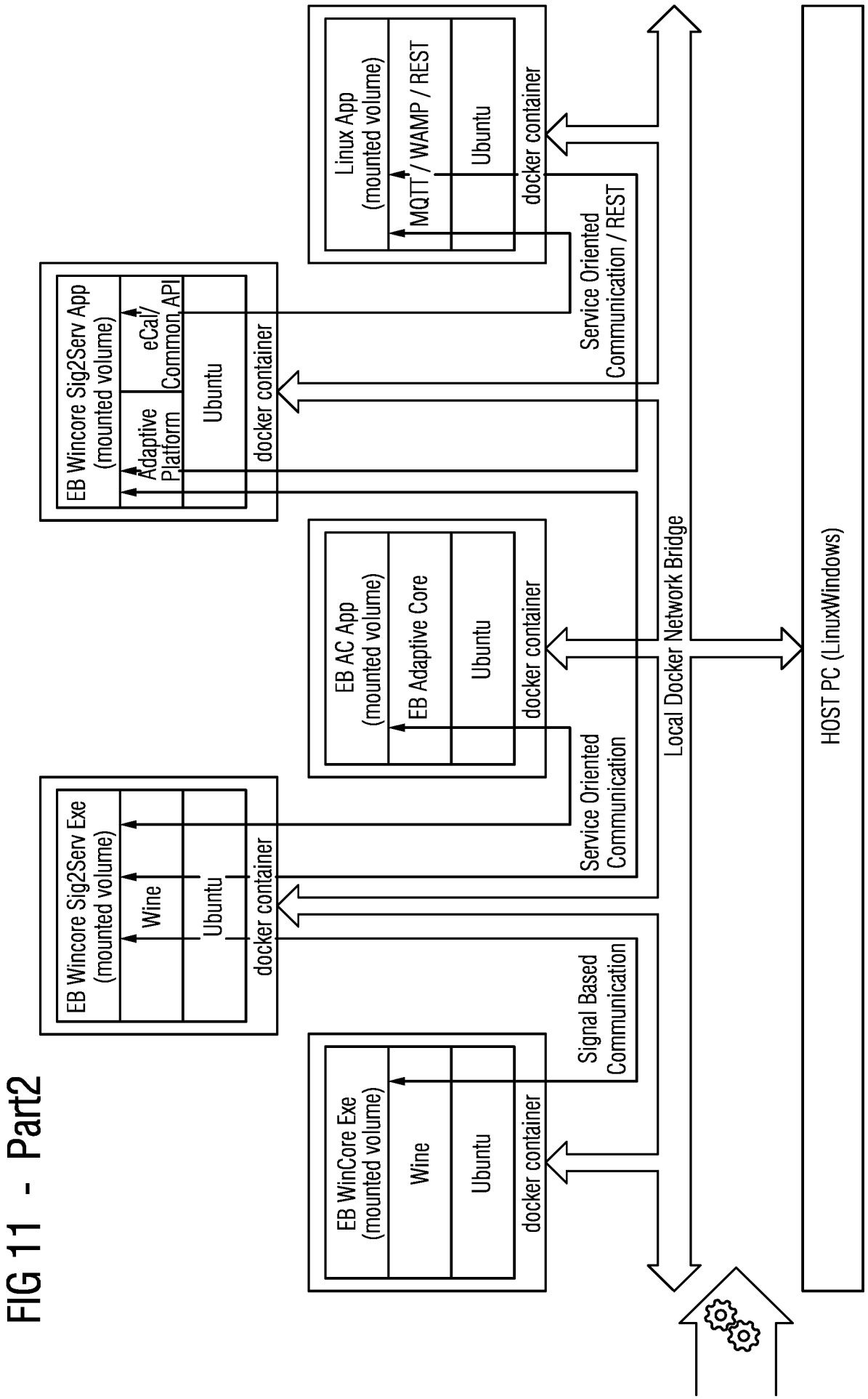


FIG 11 - Part2

