Unser Fokus:

- Den Anwender dabei unterstützen, sich auf sein eigentliches Problem konzentrieren zu können
- Ein Werkzeug an die Hand zu geben, die Use Cases umzusetzen
- Dazu zählen u.a.
  - Wiederverwendbarkeit von Komponenten
  - Plug & Play
  - Portierbarkeit auf andere Systeme / Problemfälle
  - Rapid Prototyping
  - Verwendung von Standards (CAN, CAN-FD, Flexray, XCP, SOME/IP)

Produkt muss folgendes erfüllen:

- Gleiches Verhalten
  - Online-Betrieb (LIVE/RECORDING)
  - Offline-Betrieb (PLAYBACK)
- Modularität / Erweiterbarkeit
- Performance
- Wartbarkeit
- Dokumentation
- Usability
- Robustheit

- Modern (am Stand der Technik)
- Binär-Kompatibilität innerhalb einer Major-Version
- Code-Kompatibilität innerhalb einer Major-Version

Limitierungen in ADTF 2.x

- Begleichung technischer Schuld
- Ext-Interfaces
- Wartung
- Legacy Code
- Keine API Verbesserungen (hinsichtlich Schutz vor Programmierfehlern beim Nutzer) möglich
- Plattform/Toolchain gebunden

➔ **Neue Major Version ADTF 3.x**

Weiterhin in ADTF 3.x vorhanden:

- Tooling zur Modelierung von ADTF Sessions
- adtfplugins zur Wiederverwendung
  - Filter (im Filter Graph)
  - Streaming Services (Sources/Sinks im Streaming Graph)
  - System / UI Services (Systemausprägung)
  - Jeweils durch SDK auch beim Anwender erweiterbar
- uCOM Architektur zur Kommunikation über Binärgrenzen (zwischen adtfplugins)

- Binärkompatibilität
- Codekompatibilität
- ADTF System Ausprägung und Funktionalität durch Services erweiterbar
- (austauschbares) Reference-Clock Konzept inkl. Clock Synchronisation
- Kernel Konzept für Threads und Timer
- Signal Registry/Listener mit Publish/Subscribe
- Interface Binding für Function-Calls zwischen Komponenten
- Logging Mechanismus
- XSystem zum Handling von UI Widgets
- DDL Beschreibung und Media Description Support
- Support für ADTF 2 Aufzeichnungen
- Kommunikation/Steuerung von/zu ADTF 2.x Launcher
- Library zur Behandlung von (adtf)dat files:
    - Open Source Library für (adtf)dat files (Info, Reader/Processor, eigene (De)Serialisierungen, eigene Tools/Libraries)
    - Open Source GUI Tooling für (adtf)dat files (Info, Import/Export)
- Verwendung von Standards
    - ASAM/Vector/...
    - 3rd Party Libraries (Qt, OSG, …)
    - 3rd Party Tooling (Easy Profiler, …)

Neu im Vergleich zu ADTF 2

- Auftrennung Tools, Framework und SDK

    - Headless und GUI Tools
- Konfigurationszeitpunkt von Laufzeit getrennt
- Trennung von Daten- und Kontrollfluss
- Laufzeitverhalten / Performance verbessert
    - seperate Prozesse / keine Quereffekte (z.b. Tooling und ADTF System)
    - schlanke Systemausprägung (nur das geladen, was gebraucht wird)
    - Laufzeitverhalten des Filters von außen konfigurierbar (keine internen Threads/Timer, selber Daten halten) -> keine Deadlocks
    - Optimiertes Ladeverhalten, Parser & Co
- Trennung des Graphen in Streaming Graph (Schnittstelle von/zu ADTF) und Filter Graph (Algorithmus)
    - Zusätzlicher Runlevel (z.B. für Hardware Treiber ohne neuen Init)
    - Austauschbarkeit (FG kann in LIVE und PLAYBACK SG verwendet werden)
- Getrennte APIs (Use Case basiert)
- Kommunikation über Systemgrenzen
    - RPC (Steuerung) -> Player, Recorder, ...
    - IPC (Datenaustausch) -> UDP, TCP, Pipes/Sockets, ...
- namespace-versionierte Interfaces zur API-Erweiterung (Binär-/Codekompatibel)
- Trennung Playback von Wiedergabe (Playback Service) und Ansteuerung (Playback Streaming Source) -> ADTF 2 Architekturfehler
- Redesigned Extended Data Service in Player / Recorder integriert -> Attached Files

- Unabhängiges Scripting (ADTF Control, ADTF Config Tool)
- Stream Type statt Major/Subtype
    - Erweiterbar durch Properties
    - Media Description als Teil des Streams -> Stichwort (adtf)dat file
- Plugin Description zur Beschreibung des adtfplugins
    - Erstellung beim Build möglich
    - CE muss nicht mehr adtfplugin laden
    - Abhängigkeiten definier- und automatisierbar
    - Hilfe möglich
- Exception Handling
- Scripting Filter (JavaScript / QtQuick) für rapid prototyping
- QtQuick Filter für rapid prototyping
- Gelöstes und erweiterbares Property Handling
- (besser) dokumentierte API mit u.a. const correctness, Exceptions, Fehler Callstack, Typsicherheit, Reference Counting, Templates
- Guides

Keyfacts ADTF 3.4

- Feature Complete

Keyfacts ADTF 3.5

- Redesign Filter SDK (Convenience API)

Keyfacts ADTF 3.6:

- Home View für ADTF Configuration Editor
- Redesign System Editor
- Redesign Settings Editor
- Redesign Property Handling
- Redesign Streaming/Filter Graph Editor
- Plugin Description komplett im Code beschreibbar
- Upgrade Qt 5.12.4
- Upgrade to Win 10 VS 2017 VC141
- Usability / Tool & Shortcut Support
- Erweiterung SDK Doku / Guides
- Dark Theme
- Redesign Icons
- UI Stabilität
- ADTF 2 Compressed Video Support (Decoding)
- Redesign Javascript SDK
- Shared Qt Window
- Read-Only Mode Config Editor

Ausblick (2019/2020), teilweise schon umgesetzt:

- Verteiltes System
- Substreams
    - An einen Pin werden multiple Datenströme bereitgestellt
    - Publish/Subscribe auf Empängerseite
    - Dynamisches Encoding: Nur das was angefragt wird, wird auch gesendet
- (erweiterbares) Display auf Qt3D Basis

- Usability, u.a.
    - Tooltips / Jump-2-Doku
    - Toolingunterstützte Automatismen/JavaScript Scheduler, z.b. auto-add Sample Stream
    - Redesign Property Editor
    - Redesign Session Editor
    - Copy/Paste
    - QML API für CE, u.a. Dynamic Properties
    - Component Store
    - Touch Support
- Error Handling Service
- include Mechanismus bestehender Graphen
- Runtime Property Editor GUI
- ADTF System Reporter für Support Dump
- SOME/IP Support
- Debug Tooling
    - Breakpoints
    - Sample Überwachung
- Python Scripting Filter
- Neuer Lizenzmechanismus
- DDL Mapping Filter
- PCap Anbindung
- MDF Support
- HDF Support
- MQTT Anbindung
- ADTF goes Cloud
- Autosar SecOC für gesicherte Bus-Signalübertragung