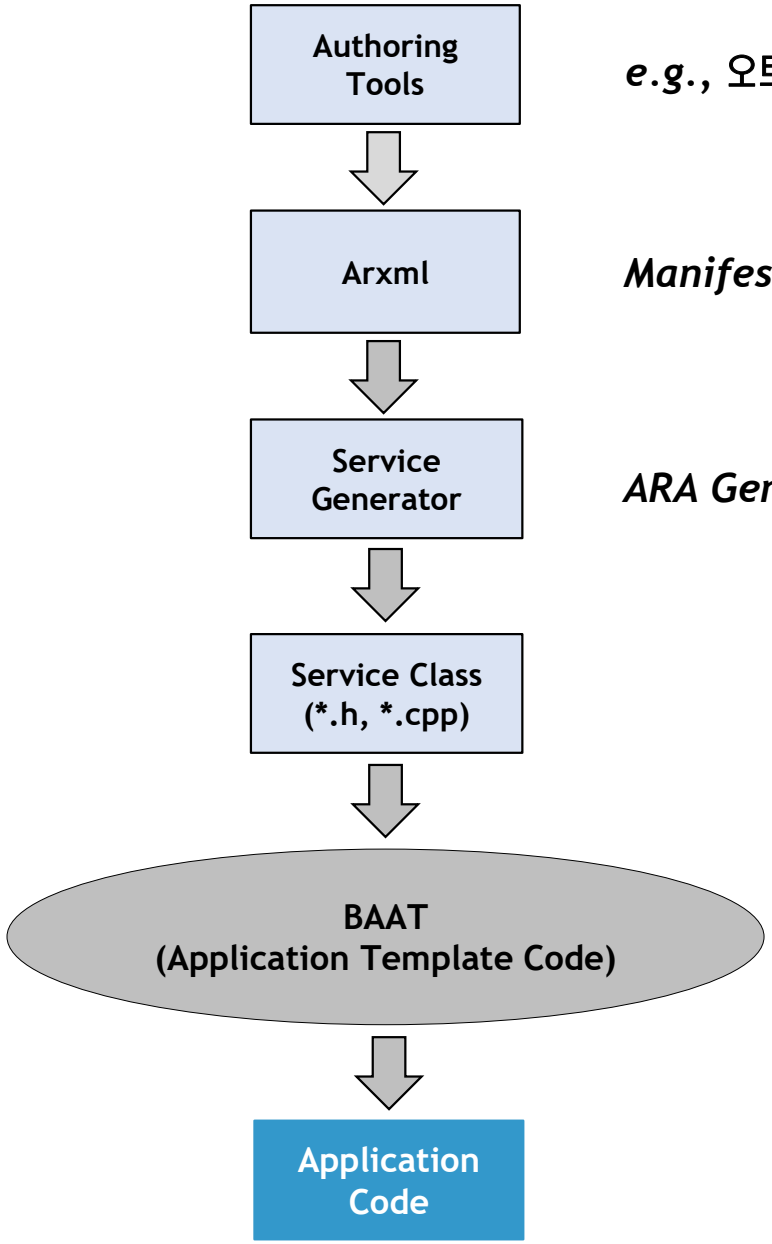
BNS AUTOSAR Adaptive Tool (BAAT)

BNS AUTOSAR Adaptive Tool (BAAT)

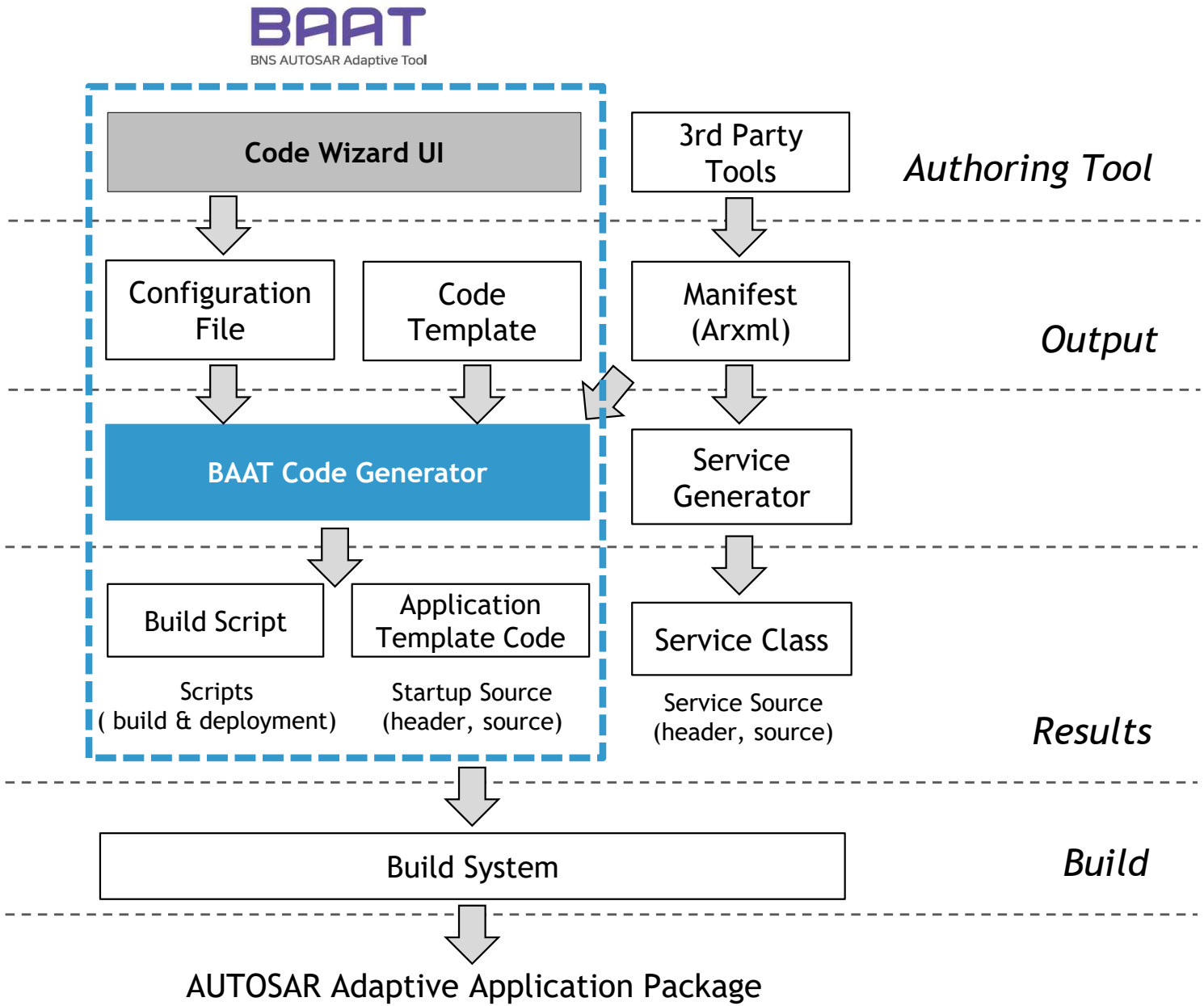


***e.g.*, 오트론 mobilgene A Studio™(ARXML 생성 tool)**

Manifest Files

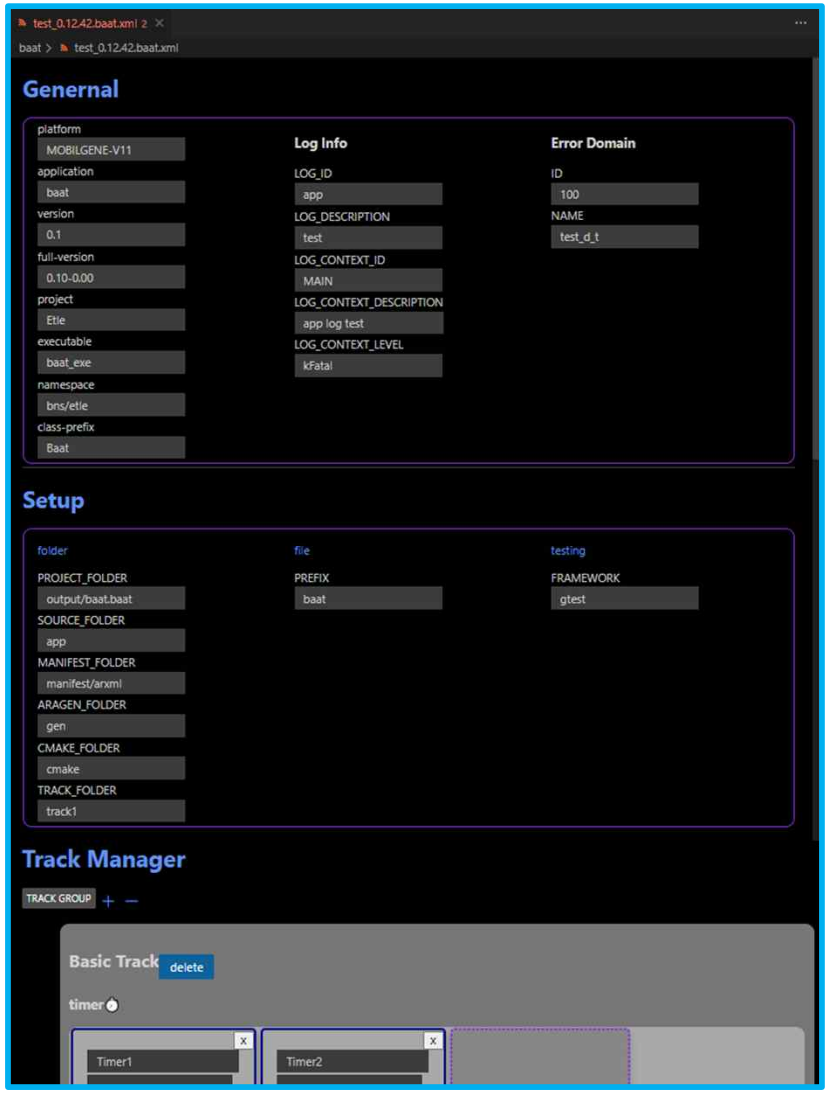
ARA Gen

BNS AUTOSAR Adaptive Tool (BAAT)



BNS AUTOSAR Adaptive Tool (BAAT)

- **Code Wizard UI: Configuration File 생성을 위한 UI 입력창**



1. GENERAL: 프로젝트 버전, 이름, 로그 정보, 에러 도메인 설정
2. SETUP: 프로젝트의 유닛 테스트 설정 및 폴더 설정
3. TRACK MANAGER: Application State 관리 정보 설정
4. SERVICE: Application이 사용하고 있는 서비스 정보 설정
5. MANIFEST: Application 만들 때 사용한 *.arxml 파일 정보
6. RESOURCES : Application이 사용하는 Resource 파일 정보
7. BUILD: 빌드 관리 정보 설정

BAAT 사용 시 장점 및 Framework

장 점	BAAT Framework 기능
• Adaptive Application 개발 경험이 없는 개발자가 쉽고 빠르고, 완성도 높은 Application 개발 • 다양한 Adaptive Application 개발 경험을 통한 레퍼런스 소스 코드 생성 • 요구 사항 변경에 대한 빠른 어플리케이션 구현 • Build용 Script 생성 • 정형화된 코드 생성을 통하여 개발자 오류를 줄임 • AUTOSAR Adaptive C++ Guideline 적용	• 서비스 Offer & Discovery • Multiple Service(Method, Event, Field 생성) • Multiple Service Instance • Thread(Service Instance 당 1개 생성) • Exception Handling • State Machine • Unit Test Code 생성(gTest, gMock) • UML Code 생성(Class Diagram)

AUTOSAR ARA Gen 생성 코드 예

service_desc_radar.cpp

```
// ------------------------------------------------------------------------------

#include "service_desc_radar.h"
namespace apd {
namespace da {
namespace radar_binding {
namespace vsomeip {
namespace descriptors {

constexpr ara::com::internal::vsomeip::types::EventGroupId FrontObjectDistance::event_groups[];
constexpr ara::com::internal::vsomeip::types::EventGroupId O
constexpr ara::com::internal::vsomeip::types::EventGroupId R
constexpr ara::com::internal::vsomeip::types::EventGroupId U
constexpr ara::com::internal::vsomeip::types::EventGroupId b
constexpr ara::com::internal::vsomeip::types::EventGroupId d
constexpr ara::com::internal::vsomeip::types::EventGroupId p

} // namespace descriptors
} // namespace vsomeip
} // namespace radar_binding
} // namespace da
} // namespace apd
```

Service Class 만 제공
빌드 및 Adaptive Application
코드 개발 필요

ara_com_main-radarservice.cpp

```
#endif // HAS_OPENDDS_BINDING

namespace ara {
namespace com {
namespace internal {
namespace runtime {

void Initialize() {
#ifdef HAS_VSOMEIP_BINDING
    vsomeip::runtime::Register();
    apd::da::radar_binding::vsomeip::registerErrors();
    ara::com::internal::vsomeip::common::error_domains::Registry::registerDomain(a
#endif // HAS_VSOMEIP_BINDING

#ifdef HAS_OPENDDS_BINDING
    dds::runtime::Register();
    ara::com::internal::dds::common::error_domains::Registry::registerDomain(ara::
#endif // HAS_OPENDDS_BINDING
}

} // namespace runtime
} // namespace internal
} // namespace com
} // namespace ara
```

BAAT 자동 생성 코드 예(1/3)

radar_main.cpp

```cpp
// ------------------------------------------------------------------------
// [BAAT-GEN] RadarMain (static)
//  Radar Main function.
// ------------------------------------------------------------------------
static void RadarMain(void) noexcept(false)
{
    RadarSig::Init();
    RadarLog::Init();

    if (ReportRunning())
    {
        ShowStartupBanner();

        RadarApp radar_app{};
        radar_app.Run();

        ShowTerminateBanner();
    }

    ReportTerminating();
}

} // namespace bns::etle

// ------------------------------------------------------------------------
// [BAAT-GEN] main
//  Main entry function.
// ------------------------------------------------------------------------
int main(int argc, char** argv)
{
    try
    {
        if (bns::etle::testing::RadarTester::Run(argc, argv))
        {
            return 0;
        }

        bns::etle::RadarMain();
        return 0;
    }
    catch (const std::exception& e)
```

빌드 및 실행 가능한
Adaptive Application 코드 생성

radar_app.cpp

```cpp
// ------------------------------------------------------------------------
// [BAAT-GEN] Run (public, virtual, noexcept)
// ------------------------------------------------------------------------
void RadarApp::Run(void) noexcept
{
    Log(__LINE__) << "Run()";

    using radarSkeletonActT = ::ara::com::sample::radarSkeletonAct;

    radarSkeletonActT radar_provider{ kInstanceSpecifier_radar_provider };

    RadarServices radar_services{ radar_provider };
    radar_services.SetUp();
    radar_services.OfferAllServices();

    while (RadarSig::IsRunning())
    {
        try
        {
            constexpr const auto kMsmAppLoopDuration_msec{ 1000LL };
            RadarSig::WaitFor(kRadarAppLoopDuration_msec);
        }
        catch (const std::exception& e)
        {
            LogFatal(__LINE__) << "Run() EXCEPTION :" << e.what();
            RequestToRestart(); // [[noreturn]]
        }
    }

    radar_services.Destroy();

    Log(__LINE__) << "Run() terminated";
}

// ------------------------------------------------------------------------
// [BAAT-GEN] SignalHandler (protected, const, noexcept)
// ------------------------------------------------------------------------
void RadarApp::SignalHandler(std::int32_t signal) const noexcept
{
```

BAAT 자동 생성 코드 예(2/3)

Service Instance Manifest(ARXML)

```
<FIELDS>
  <FIELD>
    <SHORT-NAME>FrontObjectDistance</SHORT-NAME>
    <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/AUTOSAR/StdTypes/uint16_t</TYPE-TR
    <HAS-GETTER>false</HAS-GETTER>
    <HAS-NOTIFIER>true</HAS-NOTIFIER>
    <HAS-SETTER>false</HAS-SETTER>
  </FIELD>


<METHODS>
  <CLIENT-SERVER-OPERATION>
    <SHORT-NAME>Adjust</SHORT-NAME>
    <ARGUMENTS>
      <ARGUMENT-DATA-PROTOTYPE>
        <SHORT-NAME>target_position</SHORT-NAME>
        <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/apd/da/methods/Position
        <DIRECTION>IN</DIRECTION>
      </ARGUMENT-DATA-PROTOTYPE>
      <ARGUMENT-DATA-PROTOTYPE>
        <SHORT-NAME>success</SHORT-NAME>
        <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/AUTOSAR/StdTypes/bool</TYPE-TREF>
        <DIRECTION>OUT</DIRECTION>
      </ARGUMENT-DATA-PROTOTYPE>
      <ARGUMENT-DATA-PROTOTYPE>
        <SHORT-NAME>effective_position</SHORT-NAME>
        <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/apd/da/methods/Position</TYPE-TREF>
        <DIRECTION>OUT</DIRECTION>
      </ARGUMENT-DATA-PROTOTYPE>
    </ARGUMENTS>
  </CLIENT-SERVER-OPERATION>
  <CLIENT-SERVER-OPERATION>
    <SHORT-NAME>Calibrate</SHORT-NAME>
    <ARGUMENTS>
      <ARGUMENT-DATA-PROTOTYPE>
        <SHORT-NAME>configuration</SHORT-NAME>
        <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/apd/CppImplementationDataTypes/Strin
        <DIRECTION>IN</DIRECTION>
      </ARGUMENT-DATA-PROTOTYPE>
      <ARGUMENT-DATA-PROTOTYPE>
        <SHORT-NAME>variant</SHORT-NAME>
        <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/apd/da/types/FusionVariant</TYPE-TRE
        <DIRECTION>IN</DIRECTION>
      </ARGUMENT-DATA-PROTOTYPE>
```

자동 생성 코드

```
// ------------------------------------------------------------------------------
// [BAAT-GEN] UpdateFrontObjectDistance (public, virtual, noexcept)
//  FIELD : FrontObjectDistance [notifier]
//      FrontObjectDistanceFieldT (std::uint16_t)
// ------------------------------------------------------------------------------
void radarSkeletonImp::UpdateFrontObjectDistance(const FrontObjectDistanceFieldT& value) noexcept
{
    logger_.LogDebug() << "radarSkeletonImp::UpdateFrontObjectDistance()";

    try
    {
        Front
    }
    catch (co
    {
        logge
    }
}
```

```
// ------------------------------------------------------------------------------
// [BAAT-GEN] Method : Adjust (public, virtual)
// ------------------------------------------------------------------------------
ara::core::Future<radar::AdjustOutput>
radarSkeletonImp::Adjust(const ::ara::com::sample::Position& target_position) noexcept(false)
{
    logger_.LogDebug() << "radarSkeletonImp::Adjust()";

    ara::core::Promise<radar::AdjustOutput> promise;

    try
    {
        if (skeleton_act_ptr_ != nullptr)
        {
            ara::core::Result<radar::AdjustOutput> result{ skeleton_act_ptr_->Adjust(target_position) };

            if (result.HasValue())
            {
                promise.set_value(std::move(result.Value()));
            }
            else
            {
                promise.SetError(std::move(result.Error()));
            }
        }
        else
        {
            promise.SetError(0);
        }
    }
    catch (const std::exception& e)
    {
        logger_.LogError() << "radarSkeletonImp::Adjust() EXCEPTION :" << e.what();
        promise.SetError(0);
    }

    return std::move(promise.get_future());
}
```

BAAT 자동 생성 코드 예(3/3)

Service Instance Manifest(ARXML)

```
<FIELD>
  <SHORT-NAME>RearObjectDistance</SHORT-NAME>
  <TYPE-TREF DEST="STD-CPP-IMPLEMENTATION-DATA-TYPE">/AUTOSAR
  <HAS-GETTER>true</HAS-GETTER>
  <HAS-NOTIFIER>false</HAS-NOTIFIER>
  <HAS-SETTER>false</HAS-SETTER>
</FIELD>
```

자동 생성 코드

```
// ------------------------------------------------------------------------------
// [BAAT-GEN] GetRearObjectDistance (public, virtual, noexcept)
//  FIELD : RearObjectDistance [GETTER]
//      RearObjectDistanceFieldT (std::uint16_t)
// ------------------------------------------------------------------------------
const ara::core::Result<radarProxyAct::RearObjectDistanceFieldT> radarProxyAct: GetRearObjectDistance(void) noexcept
{
    logger_.LogDebug() << "radarProxyAct::GetRearObjectDistance()";

    try
    {
        if (IsServiceAvailable())
        {
            auto future{ proxy_delegate_.GetRearObjectDistance() };
            auto status{ future.wait_for(std::chrono::milliseconds{ kradarServiceTimeout_msec }) };

            if (status == ara::core::future_status::kReady)
            {
                const auto& result{ future.GetResult() };

                if (!result.HasValue())
                {
                    logger_.LogError() << result.Error().Value() << result.Error().Message();
                }

                return result;
            }

            logger_.LogDebug() << "Timeout - radarProxyAct::GetRearObjectDistance()";
            return ara::core::Result<RearObjectDistanceFieldT>(FusionErrcT::kTimeout);
        }

        logger_.LogDebug() << "Service is unavailable";
        return ara::core::Result<RearObjectDistanceFieldT>(FusionErrcT::kServiceNotAvailable);
    }
    catch (const std::exception& e)
    {
        logger_.LogError() << "radarProxyAct::GetRearObjectDistance() EXCEPTION :" << e.what();
        return ara::core::Result<RearObjectDistanceFieldT>(FusionErrcT::kException);
    }
}
```

Exception 처리 강화

Reference 코드에서의 예외 처리(Adjust 메서드의 구현 비교)

AUTOSAR Sample Code 에서의 Adjust () 구현

```
void FusionActivity::asyncMethodCall()
{
    Position target_position;
    target_position.x = m_ud_min_500_500(m_rand_eng);
    target_position.y = m_ud_min_500_500(m_rand_eng);
    target_position.z = m_ud_min_500_500(m_rand_eng);

    m_async_count++;

    auto adjust_future = m_proxy->Adjust(target_position);

    // Polling the results.
    while (!adjust_future.is_ready()) {
        // Results are not ready yet.
        std::this_thread::sleep_for(std::chrono::milliseconds(500
        m_logger.LogInfo() << METHODS_HEADER << "Radar is adjusti
    }

    // GetResult will return immediately.
    auto r = adjust_future.GetResult();
    if (r.HasValue()) {
        auto adjust_output = r.Value();
        if (adjust_output.success) {
            m_logger.LogInfo() << METHODS_HEADER << "Adjusting po
                               << adjust_output.effective_positio

        } else {
            m_logger.LogWarn() << METHODS_HEADER
                               << "Adjusting position was not suc
                               << adjust_output.effective_positio
                               << adjust_output.effective_positio
                               << adjust_output.effective_positio
        }
    } else {
        auto err = r.Error();
        m_logger.LogWarn() << ERROR_HEADER << "Code: " << err.Val
        m_logger.LogWarn() << ERROR_HEADER << "Message: " << err.
    }
}
```

BAAT 자동 생성툴에서의 Adjust () 구현

```
// ------------------------------------------------------------------------
// [BAAT-GEN] Adjust (public, virtual, noexcept)
//  METHOD : Adjust
// ------------------------------------------------------------------------
ara::core::Result<radar::AdjustOutput>
radarProxyAct::Adjust(const ::ara::com::sample::Position& target_position) noexcept
{
    logger_.LogDebug() << "radarProxyAct::Adjust()";

    try
    {
        if (IsServiceAvailable())
        {
            auto future{ proxy_delegate_.Adjust(target_position) };
            auto status{ future.wait_for(std::chrono::milliseconds{ kradarServiceTimeout_msec }) };

            if (status == ara::core::future_status::kReady)
            {
                const auto& result{ future.GetResult() };

                if (!result.HasValue())
                {
                    logger_.LogError() << result.Error().Value() << result.Error().Message();
                }

                return result;
            }

            logger_.LogDebug() << "Timeout - radarProxyAct::Adjust()";
            return ara::core::Result<radar::AdjustOutput>(FusionErrcT::kTimeout);
        }

        logger_.LogDebug() << "Service is unavailable";
        return ara::core::Result<radar::AdjustOutput>(FusionErrcT::kServiceNotAvailable);
    }
    catch (const std::exception& e)
    {
        logger_.LogError() << "radarProxyAct::Adjust() EXCEPTION :" << e.what();
        return ara::core::Result<radar::AdjustOutput>(FusionErrcT::kException);
    }
}
```

Exception 처리 강화

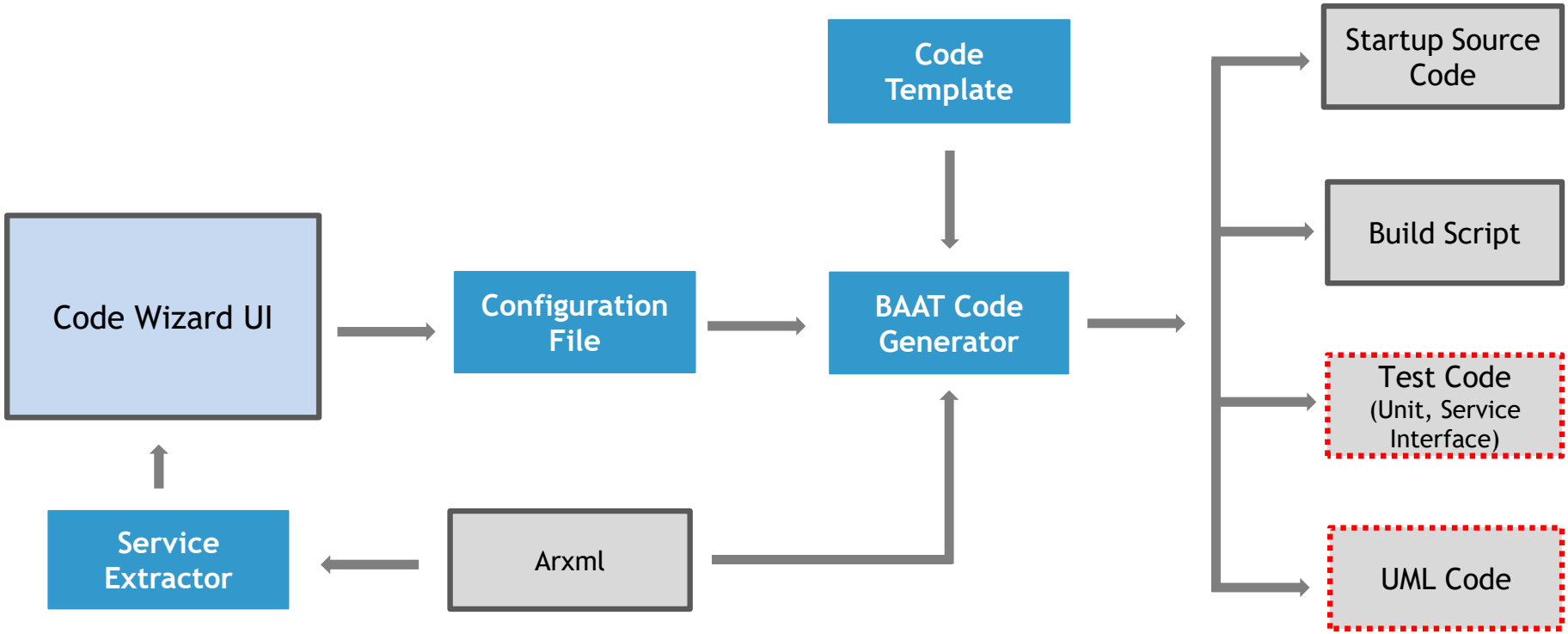
BAAT 자동 생성 파일 예 (Proxy Files)

종 류		파일명	종 류	파일명
Application Source Code		fusion_app.cpp	Application Header Code	fusion_app.h
		fusion_base.cpp		fusion_base.h
		fusion_health.cpp		fusion_health.h
		fusion_services.cpp		fusion_services.h
		fusion_sig.cpp		fusion_sig.h
		fusion_tester.cpp		fusion_tester.h
		fusion_main.cpp		fusion_err.h
		CMakeLists.txt		fusion_log.h
Track	basic track	basic_timer_def.h	Required Service Code	CMakeLists.txt
		basic_track.cpp		radar_proxy_act.cpp
		basic_track.h		radar_proxy_act.h
	extended track	extended_state_def.h		radar_proxy_delegate.cpp
		extended_timer_def.h		radar_proxy_delegate.h
		extended_track.cpp		radar_proxy_enum_log.h
		extended_track.h	Build/Deploy Script Files	build.sh
		fusion_state_track_base.h		deploy.sh
		fusion_track_base.h		
		fusion_track_common_def.h		
		fusion_track_shared_data.h		

BAAT 자동 생성 파일 예 (Skeleton Files)

종 류	파일명	종 류	파일명
Application Source Code	radar_app.cpp	Application Header Code	radar_app.h
	radar_base.cpp		radar_base.h
	radar_health.cpp		radar_health.h
	radar_services.cpp		radar_services.h
	radar_sig.cpp		radar_sig.h
	radar_tester.cpp		radar_tester.h
	radar_main.cpp		radar_err.h
	CMakeLists.txt		radar_log.h
Build/Deploy Script Files	build.sh	Provided Service Code	CMakeLists.txt
	deploy.sh		radar_skeleton_act.cpp
			radar_skeleton_act.h
			radar_skeleton_enum_log.h
			radar_skeleton_imp.cpp
			radar_skeleton_imp.h

BNS AUTOSAR Adaptive Tool (BAAT)



Unit Test 및 Service Interface Test Code

Skeleton 테스트

```
TEST(StateManager, GetNetRequest)
{
    auto* instance_ptr = statemanager_project::unittest::StateM
    ASSERT_TRUE(instance_ptr != nullptr);

    auto* logger_ptr = statemanager_project::unittest::StateManagerSkeletonGTest::LoggerPtr();
    ASSERT_TRUE(logger_ptr != nullptr);

    const std::vector<::vehicle::system::NetworkRequest> test_values{ ::vehicle::system::NetworkRequest::STARTUP,
        ::vehicle::system::NetworkRequest::SHUTDOWN };

    // clang-format off
    for (const auto& test_value : test_values)
    {
        instance_ptr->GetSkeleton().WriteNetRequest(test_value);
        SkeletonMethodTest<::vehicle::system::NetworkRequest>(
            [instance_ptr, logger_ptr]()
            {
                return instance_ptr->GetSkeleton().GetNetRequest();
            },
            [logger_ptr, test_value](::vehicle::system::NetworkRequest output)
            {
                EXPECT_EQ(test_value, output);
            });
    }
    // clang-format on
}
```

펄드 테스트 코드

Proxy 테스트

```
TEST(PackageManagement, GetCurrentStatus)
{
    auto* instance_ptr = statemanager_project::unittest::PackageManagementProxyGTest::InstancePtr();
    ASSERT_TRUE(instance_ptr != nullptr);

    auto* logger_ptr = statemanager_project::unittest::PackageManagementProxyGTest::LoggerPtr();
    ASSERT_TRUE(logger_ptr != nullptr);

    // clang-format off
    ProxyFieldGetTest<::ara::ucm::pkgmgr::PackageManagerStatusType>(
        [instance_ptr, logger_ptr]()
        {
            return instance_ptr->GetCurrentStatus();
        });
    // clang-format on
}
```

펄드 테스트 코드

```
TEST(StateManager, MethodRegisterSessionOwner)
{
    auto* instance_ptr = statemanager_project::unittest::StateManagerSkeletonGTest::InstancePtr();
    ASSERT_TRUE(instance_ptr != nullptr);

    auto* logger_ptr = statemanager_project::unittest::StateManagerSkeletonGTest::LoggerPtr();
    ASSERT_TRUE(logger_ptr != nullptr);

    // clang-format off
    const std::vector<
        std::tuple<::String>> test_values
    {
        // test_value[0]
        { "" }, // In
        // test_values[1]
        { "ABC" } // Out
    };

    for (const auto& test_value : test_values)
    {
        SkeletonMethodTest<void>(
            [instance_ptr, logger_ptr, test_value]()
            {
                const auto& param1 = std::get<0>(test_value);
                return instance_ptr->GetSkeleton().RegisterSessionOwner(param1);
            });
    }
    // clang-format on
}
```

메서드 테스트 코드

```
TEST(PackageManagement, MethodFinish)
{
    auto* instance_ptr = statemanager_project::unittest::PackageManagementProxyGTest::InstancePtr();
    ASSERT_TRUE(instance_ptr != nullptr);

    auto* logger_ptr = statemanager_project::unittest::PackageManagementProxyGTest::LoggerPtr();
    ASSERT_TRUE(logger_ptr != nullptr);

    // clang-format off
    ProxyMethodTest<void>(
        [instance_ptr, logger_ptr]()
        {
            return instance_ptr->Finish();
        });
    // clang-format on
}
```

메서드 테스트 코드

UML Code

```
class ets6 {
public:
 static constexpr ara::com::internal::ServiceId service_id = 0xde6b;
```

```
class ets6Skeleton : public bns::etle::ets6::ets6, public ara::com::internal::
public:
  /// @uptrace{SWS_CM_00130}
  ets6Skeleton(ara::com::InstanceIdentifier instance_id, ara::com::MethodCallP
  virtual ~ets6Skeleton() {
    StopOfferService();
  }
```

```
class Ets6SkeletonImp : public bns::etle::ets6::skeleton::ets6Skeleton
{
public:
```

```
class Ets6SkeletonData
{
public:
```

```
class Ets6SkeletonActBase
{
public:
    Ets6SkeletonActBase(void) noexcept(false);
```

```
template <typename SkeletonImpType_>
class Ets6SkeletonAct : public Ets6SkeletonActBase
{
public:
    Ets6SkeletonAct(const ara::com::InstanceIdentifier& instance_identifier) r
        : skeleton_{ instance_identifier, kMode, logger, data }
    {
        logger.LogDebug() << __FUNCTION__;
    }

pub
    Ets6SkeletonAct(const Ets6SkeletonAct&) = delete;
    Ets6SkeletonAct(Ets6SkeletonAct&&) = delete;

    Ets6SkeletonAct& operator=(const Ets6SkeletonAct&) = delete;
    Ets6SkeletonAct& operator=(Ets6SkeletonAct&&) = delete;
```

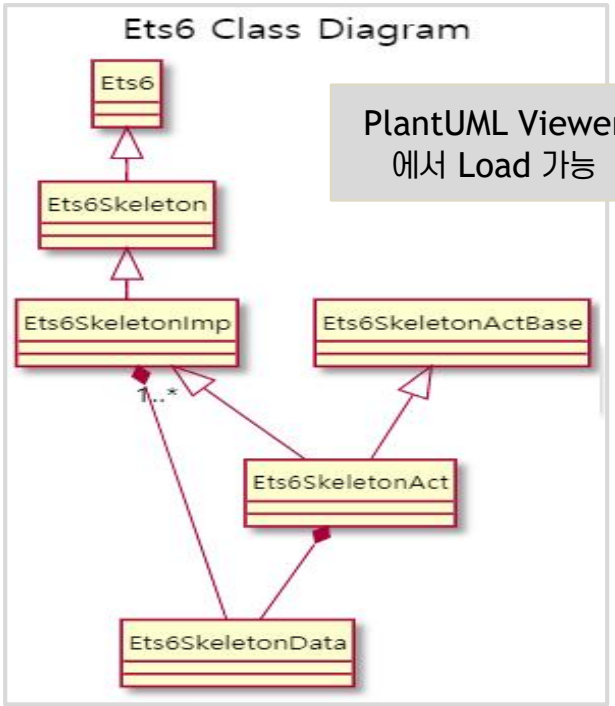
BAAT Code Generator 와 상위에서 상속받을 클래스들의 관계를 파악하여 Class Diagram Code 생성

```
@startuml
hide circle
title \n Ets6 Class Diagram \n

class Ets6
class Ets6Skeleton
class Ets6SkeletonImp
class Ets6SkeletonData
class Ets6SkeletonActBa
class Ets6SkeletonAct

Ets6 <|-- Ets6Skeleton
Ets6Skeleton <|-- Ets6SkeletonImp
Ets6SkeletonImp *-- Ets6SkeletonData
Ets6SkeletonActBase <|-- Ets6SkeletonAct
Ets6SkeletonImp "1..*"<|-- Ets6SkeletonAct
Ets6SkeletonAct *-- Ets6SkeletonData
@enduml
```

Class Diagram PlantUML Code 자동 생성 예시



PlantUML Viewer 에서 Load 가능

감사합니다.

152-790 서울시 구로구 디지털로 306 대륭2차(구로동) 702호 TEL. 02.521.1520 FAX. 02.2082.0220
www.bns.co.kr

BNSoft
Mobile Technology Leader