Coverity Connect MCP Server - クラス設計図・UMLダイアグラム

Version: 1.0.0
作成日: 2025年7月21日
更新日: 2025年7月21日

📋 概要

本ドキュメントでは、Coverity Connect MCP Serverの主要クラスとコンポーネントの設計をUMLダイアグラムで表現します。

🏗️ 1. クラス図 (Class Diagram)

```
mermaid
```

```
classDiagram
    class FastMCP {
        +String name
        +Dict tools
        +Dict resources
        +__init__(name: str)
        +tool()(func: callable)
        +resource(uri: str)(func: callable)
        +run()
    }

    class CoverityClient {
        -str host
        -int port
        -bool use_ssl
        -str username
        -str password
        -str base_url
        -Optional~ClientSession~ _session

        +__init__(host, port, use_ssl, username, password)
        +_get_session() ClientSession
        +close()
        +_make_request(method, endpoint, params, data) Dict
        +get_projects() List~Dict~
        +get_project(project_id) Optional~Dict~
        +get_streams(project_id) List~Dict~
        +get_defects(stream_id, query, filters, limit) List~Dict~
        +get_defect_details(cid) Optional~Dict~
        +get_users(disabled, include_details, locked, limit) List~Dict~
        +get_user_details(username) Optional~Dict~
        +__aenter__() CoverityClient
        +__aexit__(exc_type, exc_val, exc_tb)
    }

    class ClientSession {
        <<aiohttp>>
        +BasicAuth auth
        +ClientTimeout timeout
        +TCPConnector connector
        +Dict headers
        +request(method, url, **kwargs) Response
        +close()
    }

    class MCPServer {
```

```
    <<main.py>>
    +initialize_client() CoverityClient
    +create_server() FastMCP
    +run_server()
    +cli()
}

class MCPTools {
    <<Tools>>
    +search_defects(...) List~Dict~
    +get_defect_details(cid) Dict
    +list_projects() List~Dict~
    +list_streams(project_id) List~Dict~
    +get_project_summary(project_id) Dict
    +list_users(...) List~Dict~
    +get_user_details(username) Dict
    +get_user_roles(username) Dict
}

class MCPResources {
    <<Resources>>
    +get_project_config(project_id) str
    +get_stream_defects(stream_id) str
}

class ConfigManager {
    <<Environment>>
    +COVERITY_HOST str
    +COVERITY_PORT int
    +COVERITY_SSL bool
    +COVAUTHUSER str
    +COVAUTHKEY str
    +validate() Optional~str~
}

%% Relationships
MCPServer --> FastMCP : creates
MCPServer --> CoverityClient : manages
MCPServer --> ConfigManager : uses
FastMCP --> MCPTools : registers
FastMCP --> MCPResources : registers
MCPTools --> CoverityClient : uses
MCPResources --> CoverityClient : uses
CoverityClient --> ClientSession : manages
CoverityClient ..|> ContextManager : implements
```

🔄 2. コンポーネント図 (Component Diagram)

```
mermaid
```

```
graph TB
  subgraph "External Systems"
    Claude[Claude Desktop]
    CoverityAPI[Coverity Connect API]
  end

  subgraph "MCP Server Process"
    subgraph "Presentation Layer"
      CLI[CLI Interface]
      MCP[MCP Protocol Handler]
    end

    subgraph "Service Layer"
      Tools[MCP Tools Component]
      Resources[MCP Resources Component]
      Server[Server Management]
    end

    subgraph "Data Access Layer"
      Client[Coverity Client]
      Session[HTTP Session Manager]
      Auth[Authentication Handler]
    end

    subgraph "Infrastructure Layer"
      HTTP[aiohttp Library]
      SSL[SSL/TLS Handler]
      Logging[Logging System]
      Config[Configuration Manager]
    end
  end

  %% External Connections
  Claude <==> MCP
  Client <==> CoverityAPI

  %% Internal Connections
  CLI --> Server
  MCP --> Tools
  MCP --> Resources
  Tools --> Client
  Resources --> Client
  Server --> Client
  Client --> Session
  Client --> Auth
  Session --> HTTP
```

```
    Auth --> HTTP
    Session --> SSL
    Server --> Config
    Client --> Logging
    Server --> Logging
```

📊 3. パッケージ図 (Package Diagram)

```
mermaid
```

```
graph LR
    subgraph "coverity_mcp_server"
        subgraph "main.py"
            MCPServer[MCP Server]
            MCPTools[MCP Tools]
            MCPResources[MCP Resources]
            CLI[CLI Interface]
        end

        subgraph "coverity_client.py"
            CoverityClient[Coverity Client]
            HTTPSession[HTTP Session]
            AsyncContext[Async Context Manager]
        end

        subgraph "config.py"
            Configuration[Configuration Management]
            Environment[Environment Variables]
        end

        subgraph "tools.py"
            LegacyTools[Legacy SOAP Tools]
            Automation[Automation Pipeline]
        end

        subgraph "resources.py"
            MCPResourceHandlers[Resource Handlers]
            FileSystem[File System Access]
        end

        subgraph "prompts.py"
            PromptTemplates[Prompt Templates]
            WorkflowGuides[Workflow Guides]
        end
    end

    subgraph "External Dependencies"
        FastMCP[fastmcp]
        aiohttp[aiohttp]
        Click[click]
        SUDS[suds-community]
    end

    %% Dependencies
    MCPServer --> FastMCP
    MCPServer --> CoverityClient
MCPServer --> Configuration
CoverityClient --> aiohttp
CLI --> Click
LegacyTools --> SUDS
```

🔄 4. アクティビティ図 (Activity Diagram)

```
mermaid
```

```
flowchart TD
    Start([Start MCP Server]) --> LoadConfig{Load Configuration}

    LoadConfig -->|Success| InitClient[Initialize Coverity Client]
    LoadConfig -->|Failure| ConfigError[Configuration Error]
    ConfigError --> End([Exit with Error])

    InitClient --> CreateSession[Create HTTP Session]
    CreateSession --> SetupSSL[Setup SSL Context]
    SetupSSL --> SetupAuth[Setup Authentication]
    SetupAuth --> CreateMCP[Create FastMCP Server]

    CreateMCP --> RegisterTools[Register MCP Tools]
    RegisterTools --> RegisterResources[Register MCP Resources]
    RegisterResources --> StartServer[Start MCP Server]

    StartServer --> WaitRequest{Wait for Request}

    WaitRequest -->|MCP Tool Call| ProcessTool[Process Tool Request]
    WaitRequest -->|MCP Resource| ProcessResource[Process Resource Request]
    WaitRequest -->|Server Shutdown| Cleanup[Cleanup Resources]

    ProcessTool --> ValidateParams{Validate Parameters}
    ValidateParams -->|Valid| CallCoverity[Call Coverity API]
    ValidateParams -->|Invalid| ReturnError[Return Error]

    CallCoverity --> ProcessResponse{Process Response}
    ProcessResponse -->|Success| FormatResponse[Format Response]
    ProcessResponse -->|Error| HandleError[Handle Error]

    FormatResponse --> ReturnResult[Return Result]
    HandleError --> ReturnError
    ReturnError --> WaitRequest
    ReturnResult --> WaitRequest

    ProcessResource --> LoadResource[Load Resource Data]
    LoadResource --> FormatResource[Format Resource]
    FormatResource --> ReturnResource[Return Resource]
    ReturnResource --> WaitRequest

    Cleanup --> CloseSession[Close HTTP Session]
    CloseSession --> End([Exit])
```

🏛️ 5. デプロイメント図 (Deployment Diagram)

mermaid

```
graph TB
  subgraph "Development Environment"
    subgraph "Developer Machine"
      DevClaude[Claude Desktop]
      DevMCP[MCP Server Process]
      MockServer[Mock Coverity Server]
    end
    DevClaude -.-> DevMCP
    DevMCP -.-> MockServer
  end

  subgraph "Production Environment"
    subgraph "Client Workstation"
      ProdClaude[Claude Desktop]
    end

    subgraph "Application Server"
      ProdMCP[MCP Server]
      SystemD[systemd Service]
      Logs[Log Files]
    end

    subgraph "Enterprise Network"
      Proxy[Corporate Proxy]
      Firewall[Firewall]
    end

    subgraph "Coverity Infrastructure"
      CoverityConnect[Coverity Connect Server]
      Database[(Analysis Database)]
    end

    ProdClaude --> ProdMCP
    SystemD --> ProdMCP
    ProdMCP --> Logs
    ProdMCP --> Proxy
    Proxy --> Firewall
    Firewall --> CoverityConnect
    CoverityConnect --> Database
  end

  subgraph "Container Environment"
    subgraph "Docker Container"
      ContainerMCP[MCP Server]
      PythonRuntime[Python 3.11 Runtime]
    end
    subgraph "Docker Host"
        DockerEngine[Docker Engine]
        EnvFiles[Environment Files]
    end

    DockerEngine --> ContainerMCP
    EnvFiles --> ContainerMCP
    ContainerMCP --> PythonRuntime
end
```

📈 6. 状態図 (State Diagram)

```
mermaid
```

```
stateDiagram-v2
    [*] --> Initializing

    Initializing --> ConfigLoading : Load environment variables
    ConfigLoading --> ConfigError : Missing required config
    ConfigLoading --> ClientInitializing : Config valid

    ConfigError --> [*] : Exit with error

    ClientInitializing --> SessionCreating : Create HTTP client
    SessionCreating --> SessionError : Connection failed
    SessionCreating --> ServerStarting : Session ready

    SessionError --> Retrying : Retry connection
    Retrying --> SessionCreating : Retry attempt
    Retrying --> [*] : Max retries exceeded

    ServerStarting --> Ready : MCP server started

    Ready --> ProcessingRequest : Receive MCP request
    Ready --> Shutdown : Shutdown signal

    ProcessingRequest --> ValidatingRequest : Parse request
    ValidatingRequest --> RequestError : Invalid request
    ValidatingRequest --> CallingAPI : Valid request

    RequestError --> Ready : Return error response

    CallingAPI --> APISuccess : API call successful
    CallingAPI --> APIError : API call failed

    APISuccess --> FormattingResponse : Process response data
    APIError --> ErrorHandling : Handle API error

    FormattingResponse --> Ready : Return formatted response
    ErrorHandling --> Ready : Return error response

    Shutdown --> Cleanup : Close sessions
    Cleanup --> [*] : Exit gracefully

    state ConfigLoading {
        [*] --> CheckingHost : Check COVERITY_HOST
        CheckingHost --> CheckingUser : Host found
        CheckingUser --> CheckingKey : User found
        CheckingKey --> [*] : Key found
        CheckingHost --> [*] : Host missing
    CheckingUser --> [*] : User missing
    CheckingKey --> [*] : Key missing
}

state ProcessingRequest {
    [*] --> ToolRequest : MCP Tool call
    [*] --> ResourceRequest : MCP Resource call
    ToolRequest --> [*] : Tool processed
    ResourceRequest --> [*] : Resource processed
}
```

🔗 7. シーケンス図 - オブジェクト間インタラクション

```
mermaid
```

```
sequenceDiagram
    participant M as main.py
    participant F as FastMCP
    participant C as CoverityClient
    participant S as aiohttp.ClientSession
    participant A as Coverity API

    Note over M,A: MCP Server Initialization

    M->>M: initialize_client()
    M->>C: CoverityClient(host, port, ssl, user, pass)
    C->>C: __init__()
    C-->>M: client instance

    M->>F: FastMCP("Coverity Connect MCP Server")
    F-->>M: mcp instance

    M->>F: @mcp.tool() decorators
    M->>F: @mcp.resource() decorators
    F-->>M: tools and resources registered

    Note over M,A: Request Processing Flow

    F->>M: search_defects(severity="High")
    M->>M: initialize_client()
    M->>C: get_defects(filters={'severity': 'High'})

    C->>C: _get_session()
    alt Session not exists
        C->>S: aiohttp.ClientSession(auth, timeout, ssl)
        S-->>C: session instance
    end

    C->>C: _make_request("GET", "/api/viewContents/issues/v1", params)
    C->>S: session.request("GET", url, params=params)
    S->>A: HTTP GET with Basic Auth
    A-->>S: HTTP 200 + JSON data
    S-->>C: response
    C->>C: response.json()
    C-->>M: defects list
    M-->>F: formatted response
```

🔧 8. オブジェクトライフサイクル図

```
mermaid
```

```
sequenceDiagram
    participant OS as Operating System
    participant P as Python Process
    participant M as MCPServer
    participant C as CoverityClient
    participant S as HTTPSession

    Note over OS,S: Object Creation Phase

    OS->>P: Start Python process
    P->>M: Import main.py
    M->>M: Global variables initialization

    Note over M: coverity_client = None

    P->>M: cli() or run_server()
    M->>M: initialize_client()

    alt First call
        M->>C: CoverityClient.__init__()
        C->>C: Set instance variables
        C->>C: self._session = None
        C-->>M: client instance
        M->>M: coverity_client = client (global)
    else Subsequent calls
        M->>M: return existing coverity_client
    end

    Note over OS,S: Active Usage Phase

    loop For each MCP request
        M->>C: API method call
        C->>C: _get_session()

        alt Session doesn't exist
            C->>S: aiohttp.ClientSession()
            S->>S: Initialize connection pool
            S->>S: Setup SSL context
            S->>S: Setup authentication
            S-->>C: session ready
            C->>C: self._session = session
        else Session exists
            C->>C: return self._session
        end

        C->>S: make HTTP request
        S-->>C: response
        C-->>M: processed data
    end

    Note over OS,S: Cleanup Phase

    alt Graceful shutdown
        M->>C: close() or __aexit__()
        C->>S: session.close()
        S->>S: Close all connections
        S->>S: Cleanup resources
        S-->>C: cleanup complete
        C-->>M: cleanup complete
    else Process termination
        OS->>P: SIGTERM/SIGKILL
        P->>S: Automatic cleanup
        S->>S: Force close connections
    end

    P-->>OS: Process exit
```

🎯 9. 設計パターン適用図

```
mermaid
```

```
classDiagram
    class Singleton {
        <<pattern>>
        -_instance: CoverityClient
        +getInstance(): CoverityClient
    }

    class ContextManager {
        <<pattern>>
        +__aenter__(): self
        +__aexit__(exc_type, exc_val, exc_tb): None
    }

    class AsyncContextManager {
        <<pattern>>
        +async __aenter__(): self
        +async __aexit__(exc_type, exc_val, exc_tb): None
    }

    class Decorator {
        <<pattern>>
        +@mcp.tool()
        +@mcp.resource()
        +@click.command()
    }

    class Factory {
        <<pattern>>
        +create_server(): FastMCP
        +_get_session(): ClientSession
    }

    class Strategy {
        <<pattern>>
        +DummyDataStrategy
        +RealAPIStrategy
        +ErrorHandlingStrategy
    }

    class Observer {
        <<pattern>>
        +LoggingObserver
        +MetricsObserver
    }

    class MCPServer {
```

```
    +initialize_client()
    +create_server()
  }

  class CoverityClient {
    -_session: Optional~ClientSession~
    +_get_session()
    +close()
    +__aenter__()
    +__aexit__()
  }

  %% Pattern Implementations
  MCPServer ..|> Singleton : implements (global client)
  CoverityClient ..|> AsyncContextManager : implements
  CoverityClient ..|> Factory : implements (session creation)
  MCPServer ..|> Factory : implements (server creation)
  CoverityClient ..|> Strategy : implements (data strategies)
  MCPServer ..|> Observer : implements (logging)

  %% Pattern Usage
  Singleton --> CoverityClient : manages instance
  ContextManager --> CoverityClient : async cleanup
  Decorator --> MCPServer : tool registration
  Factory --> CoverityClient : session creation
```

🔄 10. データフロー図

```
mermaid
```

```
flowchart LR
    subgraph "Input Layer"
        UserInput[User Natural Language Input]
        CLIArgs[CLI Arguments]
        EnvVars[Environment Variables]
    end

    subgraph "Processing Layer"
        Claude[Claude Desktop]
        MCPProtocol[MCP Protocol Handler]
        RequestParser[Request Parser]
        Validator[Parameter Validator]
    end

    subgraph "Business Logic Layer"
        ToolDispatcher[Tool Dispatcher]
        ResourceHandler[Resource Handler]
        BusinessLogic[Business Logic]
        DataAggregator[Data Aggregator]
    end

    subgraph "Data Access Layer"
        HTTPClient[HTTP Client]
        SessionManager[Session Manager]
        AuthHandler[Authentication Handler]
        ErrorHandler[Error Handler]
    end

    subgraph "External Systems"
        CoverityAPI[Coverity Connect API]
        MockServer[Mock Server]
        FileSystem[File System]
    end

    subgraph "Output Layer"
        ResponseFormatter[Response Formatter]
        JSONSerializer[JSON Serializer]
        LogOutput[Log Output]
        UserResponse[User Response]
    end

    %% Data Flow
    UserInput --> Claude
    CLIArgs --> MCPProtocol
    EnvVars --> MCPProtocol
    Claude --> MCPProtocol
    MCPProtocol --> RequestParser
    RequestParser --> Validator

    Validator --> ToolDispatcher
    Validator --> ResourceHandler
    ToolDispatcher --> BusinessLogic
    ResourceHandler --> BusinessLogic
    BusinessLogic --> DataAggregator

    DataAggregator --> HTTPClient
    HTTPClient --> SessionManager
    SessionManager --> AuthHandler
    SessionManager --> ErrorHandler

    AuthHandler --> CoverityAPI
    ErrorHandler --> MockServer
    BusinessLogic --> FileSystem

    CoverityAPI --> HTTPClient
    MockServer --> HTTPClient
    FileSystem --> BusinessLogic

    HTTPClient --> DataAggregator
    DataAggregator --> ResponseFormatter
    ResponseFormatter --> JSONSerializer
    ResponseFormatter --> LogOutput

    JSONSerializer --> UserResponse
    LogOutput --> UserResponse
    UserResponse --> Claude
```

📊 11. メトリクス・監視ポイント図

```
mermaid
```

```
mindmap
  root((Monitoring & Metrics))
    Application Metrics
      Request Count
        Tool Calls per Hour
        Resource Access per Hour
        Error Rate per Tool
      Response Times
        Average Response Time
        95th Percentile Response Time
        Timeout Occurrences
      Business Metrics
        Projects Accessed
        Defects Analyzed
        Users Queried
        Most Used Tools

    System Metrics
      Resource Usage
        CPU Utilization
        Memory Usage
        Network I/O
        Disk I/O
      Connection Metrics
        Active HTTP Sessions
        Connection Pool Size
        SSL Handshake Time
        DNS Resolution Time

    Error Metrics
      HTTP Errors
        4xx Client Errors
        5xx Server Errors
        Network Timeouts
        SSL Certificate Errors
      Application Errors
        Authentication Failures
        Configuration Errors
        Data Processing Errors
        MCP Protocol Errors

    Performance Metrics
      Throughput
        Requests per Second
        Concurrent Users
        Data Transfer Rate
```

Latency
  API Response Time
  Database Query Time
  Cache Hit Rate

🔐 12. セキュリティアーキテクチャ図

mermaid

```
graph TB
  subgraph "Security Layers"
    subgraph "Authentication Layer"
      EnvVars[Environment Variables]
      BasicAuth[HTTP Basic Auth]
      Credentials[Credential Management]
    end

    subgraph "Transport Security Layer"
      SSL[SSL/TLS Encryption]
      CertValidation[Certificate Validation]
      SecureHeaders[Secure HTTP Headers]
    end

    subgraph "Application Security Layer"
      InputValidation[Input Validation]
      OutputSanitization[Output Sanitization]
      ErrorHandling[Secure Error Handling]
    end

    subgraph "Infrastructure Security Layer"
      ProcessIsolation[Process Isolation]
      FilePermissions[File Permissions]
      NetworkSecurity[Network Security]
    end
  end

  subgraph "Security Controls"
    AccessControl[Access Control]
    AuditLogging[Audit Logging]
    SecretManagement[Secret Management]
    SessionManagement[Session Management]
  end

  subgraph "Threat Mitigation"
    RateLimiting[Rate Limiting]
    TimeoutProtection[Timeout Protection]
    InjectionPrevention[Injection Prevention]
    DataEncryption[Data Encryption]
  end

  %% Security Flow
  EnvVars --> Credentials
  Credentials --> BasicAuth
  BasicAuth --> SSL
  SSL --> CertValidation
```

```
    CertValidation --> SecureHeaders

    SecureHeaders --> InputValidation
    InputValidation --> OutputSanitization
    OutputSanitization --> ErrorHandling

    ErrorHandling --> ProcessIsolation
    ProcessIsolation --> FilePermissions
    FilePermissions --> NetworkSecurity

    %% Control Integration
    AccessControl --> BasicAuth
    AuditLogging --> ErrorHandling
    SecretManagement --> Credentials
    SessionManagement --> SSL

    %% Threat Protection
    RateLimiting --> InputValidation
    TimeoutProtection --> SessionManagement
    InjectionPrevention --> InputValidation
    DataEncryption --> SSL
```

このクラス設計図・UMLダイアグラム集は、Coverity Connect MCP Serverの設計構造を多角的に表現しています。クラス関係、コンポーネント構成、処理フロー、デプロイメント構成、セキュリティアーキテクチャなど、システムの全体像を理解するための包括的な視覚化を提供しています。