# Example: Integrating PlantUML Diagrams into Architecture Documentation



## Architecture Diagrams

**Purpose:** Demonstrate how to replace ASCII box diagrams with PlantUML-generated images for high-quality PDF output

---

## Original ASCII Diagram (Markdown Viewing)

In `01_system_overview.md`, we have this ASCII diagram:

```
                    User Interface (PyQt6)

  Subject       Live       Treatment     Safety
Selection      Video       Control      Status


                            ↕

            Application Core (Business Logic)

  Session Manager        Safety Manager
```

**Problem:** This renders poorly in PDF (font issues, alignment problems, character encoding).

---

# Solution: Use PlantUML-Generated Image

## Option 1: Conditional Rendering (Recommended)

Keep both ASCII (for quick markdown viewing) and image (for PDF generation):

```
## High-Level Architecture

<!-- BEGIN ASCII DIAGRAM (Markdown viewing) -->
<details>
<summary>ASCII Diagram (click to expand)</summary>

                  User Interface (PyQt6)

 Subject       Live       Treatment     Safety
Selection      Video       Control      Status



</details>
<!-- END ASCII DIAGRAM -->

<!-- BEGIN IMAGE (PDF generation) -->
![TOSCA Container Architecture](diagrams/output/png/TOSCA%20Container%20Diagram.png)

**Figure 1:** TOSCA system architecture showing three layers: UI (PyQt6), Application Core (business
logic), and Hardware Abstraction Layer (HAL).
<!-- END IMAGE -->

The system is organized into three layers:

1. **User Interface Layer (PyQt6):** Subject selection, live video, treatment controls, safety status
2. **Application Core:** Session management, safety coordination, protocol execution, image processing
3. **Hardware Abstraction Layer:** Laser driver (Arroyo), actuator (Xeryon), camera (VmbPy), GPIO safety
interlocks (Arduino)
```

**Result:** - GitHub/text editors show ASCII diagram (quick viewing) - PDF generation uses high-quality PlantUML image - No loss of information

---

## Option 2: Direct Image Replacement (Simplest)

Replace ASCII with image reference only:

```
## High-Level Architecture

![TOSCA Container Architecture](diagrams/output/png/TOSCA%20Container%20Diagram.png)

**Figure 1:** TOSCA system architecture showing three layers: UI, Core, and HAL.

The system is organized into three layers:

1. **User Interface Layer (PyQt6):** Subject selection, live video, treatment controls, safety status
2. **Application Core:** Session management, safety coordination, protocol execution, image processing
3. **Hardware Abstraction Layer:** Laser driver (Arroyo), actuator (Xeryon), camera (VmbPy), GPIO safety
       interlocks (Arduino)
```

**Result:** - Clean, professional appearance in both markdown and PDF - Images render perfectly with pandoc - Simpler to maintain (single source of truth)

---

## Option 3: SVG for Web/Scaling

Use SVG instead of PNG for better scaling:

```
## High-Level Architecture

![TOSCA Container Architecture](diagrams/output/svg/TOSCA%20Container%20Diagram.svg)

**Figure 1:** TOSCA system architecture showing three layers: UI, Core, and HAL.
```

**Note:** SVG works well for web documentation but may have compatibility issues with some PDF engines. Use PNG for PDF generation if SVG causes problems.

---

# Mapping ASCII Diagrams to PlantUML Images

### File: `01_system_overview.md`

**Section:** High-Level Architecture - **ASCII Diagram:** 3-layer system architecture - **PlantUML Image:** `TOSCA Container Diagram.png` - **Description:** Container-level C4 diagram showing UI, Core, and HAL layers

### File: `02_database_schema.md`

**Section:** Database Schema - **ASCII Diagram:** Entity-relationship diagram - **PlantUML Image:** `TOSCA Database Schema ERD.png` - **Description:** Complete database schema with relationships

### File: `03_safety_system.md`

**Section:** Interlock Architecture - **ASCII Diagram:** Safety interlock hierarchy - **PlantUML Image:** `TOSCA Component Diagram - Application Core.png` - **Description:** Safety Manager component and interlock coordination

### File: `04_treatment_protocols.md`

**Section:** Protocol Execution Flow - **ASCII Diagram:** Treatment workflow sequence - **PlantUML Image:** `TOSCA Treatment Workflow Sequence.png` - **Description:** Sequence diagram showing treatment execution steps

---

# Image Path Guidelines

## Relative Paths (Recommended)

Use relative paths from the markdown file location:

```
![Diagram](diagrams/output/png/diagram.png)
```

**Pandoc Resource Path:**

```
pandoc file.md -o file.pdf \
  --resource-path=".:diagrams/output/png:diagrams/output/svg"
```

## Absolute Paths (Alternative)

Use absolute paths if relative paths cause issues:

```
![Diagram](/mnt/c/Users/wille/Desktop/TOSCA-dev/docs/architecture/diagrams/output/png/diagram.png)
```

**Note:** Absolute paths reduce portability. Use relative paths when possible.

## URL-Encoded Spaces

Some tools require URL-encoded spaces in filenames:

```
![Diagram](diagrams/output/png/TOSCA%20System%20Context.png)
```

**Test both formats:** - `TOSCA System Context.png` (plain) - `TOSCA%20System%20Context.png` (URL-encoded)

---

# Testing Image Integration

## Test 1: Verify Image Exists

```
cd /docs/architecture
ls -la diagrams/output/png/TOSCA*.png
```

Expected output:

```
TOSCA Component Diagram - Application Core.png
TOSCA Component Diagram - Hardware Abstraction Layer.png
TOSCA Container Diagram.png
TOSCA Data Architecture.png
TOSCA Data Flow Diagram.png
TOSCA Database Schema ERD.png
TOSCA System Context.png
TOSCA Treatment Workflow Sequence.png
```

## Test 2: Markdown Preview

Open markdown file in VS Code or GitHub to verify image renders:

```
code 01_system_overview.md
```

## Test 3: PDF Generation

Convert to PDF and verify image quality:

```
./convert-to-pdf.sh 01_system_overview.md
```

Open generated PDF:

```
xdg-open 01_system_overview.pdf   # Linux
open 01_system_overview.pdf       # macOS
start 01_system_overview.pdf      # Windows
```

---

# Image Quality Comparison

## ASCII Diagram in PDF (Current)

- Poor font rendering (requires specific Unicode fonts)
- Alignment issues (monospace font inconsistencies)
- Character encoding problems (box drawing characters may appear as squares)
- Unprofessional appearance for medical device documentation

## PlantUML Image in PDF (Recommended)

- High-resolution PNG (300 DPI recommended)
- Professional appearance
- Perfect alignment and scaling
- Consistent rendering across all PDF viewers
- Suitable for regulatory submissions (FDA, ISO 13485)

---

# Best Practices

### 1. Always Include Alt Text

Provide descriptive alt text for accessibility:

```
![TOSCA Container Diagram showing three-layer architecture: UI (PyQt6), Core (business logic), and HAL
        (hardware drivers)](diagrams/output/png/TOSCA%20Container%20Diagram.png)
```

### 2. Add Figure Captions

Use bold text below images for captions:

```
![TOSCA Container Diagram](diagrams/output/png/TOSCA%20Container%20Diagram.png)

**Figure 1:** TOSCA system architecture with three layers: UI, Core, and HAL.
```

### 3. Reference Figures in Text

Link to figures using markdown anchors:

```
As shown in [Figure 1](#figure-1-tosca-container-diagram), the system consists of three layers...

### Figure 1: TOSCA Container Diagram
![TOSCA Container Diagram](diagrams/output/png/TOSCA%20Container%20Diagram.png)
```

### 4. Maintain Diagram Sources

Always keep PlantUML sources (`.puml` files) version-controlled:

```
git add docs/architecture/diagrams/*.puml
git add docs/architecture/diagrams/output/png/*.png
git commit -m "docs: update container diagram with new components"
```

### 5. Regenerate Images After Changes

Update images when PlantUML sources change:

```
cd docs/architecture/diagrams
./generate_diagrams.sh
```

---

## Summary

**Recommended Approach:** 1. Replace ASCII diagrams with PlantUML-generated PNG images 2. Use relative paths: `diagrams/output/png/diagram.png` 3. Add descriptive alt text and figure captions 4. Use XeLaTeX or wkhtmltopdf with CSS for PDF generation 5. Keep PlantUML sources version-controlled

**Benefits:** - Professional-quality PDF output - Consistent rendering across all platforms - Suitable for medical device regulatory submissions - Maintainable diagram-as-code workflow

**Next Steps:** 1. Review `PDF_GENERATION_GUIDE.md` for detailed instructions 2. Run `./convert-to-pdf.sh` to test with sample file 3. Run `./convert-all-to-pdf.sh` to generate complete PDF documentation set

---

**Document Version:** 1.0 **Last Updated:** 2025-11-05 **Author:** Documentation Team