Issue #6: 📋 Requirements Definition Canvas

Repository: `CherrelleTucker/codesign-toolkit` **URL:** https://github.com/CherrelleTucker/codesign-toolkit/issues/6 **Author:** @CherrelleTucker

State: `open` **Labels:** ⚙️ technical-codev, 🎨 phase-co-creation, 🌱 difficulty-beginner **Assignees:** None

Created: 2025-11-14T06:22:58Z **Last Updated in GitHub:** 2025-11-17T05:57:32Z **Worksheet Version:** 2025-11-17T05:57:42.535Z

📋 Requirements Definition Canvas

Tool Category: Technical Co-Development | **Phase**: Co-Creation | **Difficulty**: 🌱 Beginner

> *Collaboratively define what your Earth observation solution needs to do, how it should work, and what success looks like.*

📋 Tool Summary Card

Attribute	Value
🎯 **Purpose**	Co-define functional requirements, interface logic, & output formats with users
⏱️ **Time Required**	2-3 hours workshop (can split into 2 sessions)
👥 **Participants**	3-6 people: end users + technical team members
📊 **Outputs**	Documented requirements, success criteria, technical specifications
🔄 **Frequency**	Once per solution, revisit when scope changes significantly
💼 **Materials**	Canvas template, sticky notes, requirements validation checklist

---

🎯 When to Use This Tool

✅ Perfect For:

- Translating user needs into technical specifications
- Bridging communication between users and developers
- Preventing scope creep and misaligned expectations
- Creating shared understanding of solution functionality

⚠️ Consider Alternatives When:

- Requirements are extremely complex (use staged approach)
- Users and technical teams can't meet simultaneously
- Project is in very early discovery phase (do stakeholder mapping first)

📋 Prerequisites & Preparation

Must Have Before Starting:

- ☐ Basic understanding of user needs (from interviews or surveys)
- ☐ Identified key users willing to participate actively
- ☐ Technical team members who understand implementation constraints
- ☐ Clear project scope and objectives

Helpful to Have:

- Existing user workflows documented
- Current tools/systems that users employ
- Any technical constraints or requirements already known
- Examples of similar solutions (for reference, not copying)

🛠️ Implementation Guide

Canvas Structure Overview

The Requirements Definition Canvas has 6 key sections:

```
┌─────────────────┬─────────────────┬─────────────────┐
│  👥 USER NEEDS  │  🎯 SOLUTION    │  📊 SUCCESS     │
│                 │  CONCEPT        │  CRITERIA       │
│ What problems   │ High-level      │ How we'll know  │
│ are we solving? │ approach        │ it's working    │
├─────────────────┼─────────────────┼─────────────────┤
│  ⚙️ FUNCTIONAL  │  🖥️ INTERFACE   │  📋 TECHNICAL   │
│  REQUIREMENTS   │  REQUIREMENTS   │  CONSTRAINTS    │
│                 │                 │                 │
│ What must it do?│ How users       │ What limits     │
│                 │ interact        │ do we have?     │
└─────────────────┴─────────────────┴─────────────────┘
```

Workshop Agenda (2.5 hours)

🎯 Setup & Context (15 minutes)

- Introductions and role clarification
- Review project background and user research findings
- Explain canvas process and expected outcomes
- Establish ground rules for collaboration

👥 Define User Needs (30 minutes)

- **Individual brainstorm (10 min)**: Each person writes user problems/needs
- **Group discussion (15 min)**: Share and cluster similar needs
- **Prioritization (5 min)**: Identify top 3-5 most critical needs

Example outputs: "Real-time wildfire detection alerts," "Integration with existing dispatch systems," "Accuracy sufficient for evacuation decisions"

🎯 Solution Concept (25 minutes)

- **Concept sketching (15 min)**: How might we address these needs?
- **Group synthesis (10 min)**: Combine ideas into coherent approach

Example output: "Automated alert system that processes satellite data every 15 minutes and sends notifications through existing emergency management channels"

⚙️ Functional Requirements (40 minutes)

- **Core functions (20 min)**: What must the solution DO?
- **Data requirements (10 min)**: What data does it need as input/output?
- **Integration needs (10 min)**: How does it connect to existing systems?

Example outputs: "Process Landsat and MODIS thermal data," "Generate confidence scores for each alert," "Export to IPAWS format"

🖥️ Interface Requirements (30 minutes)

- **User workflows (20 min)**: Step-by-step how users will interact
- **Output formats (10 min)**: What do users need to receive and when?

Example outputs: "Dashboard shows active alerts on map," "Email notifications include coordinates and confidence level," "Mobile-friendly for field personnel"

📋 Technical Constraints & Success Criteria (25 minutes)

- **Constraints (15 min)**: What technical, organizational, or resource limits exist?
- **Success criteria (10 min)**: How will we measure if this works?

Example outputs: "Must work with existing ArcGIS infrastructure," "95% uptime requirement," "Reduce false positive rate to <10%"

🔄 Review & Validation (15 minutes)

- Walk through complete canvas
- Identify any gaps or conflicts
- Confirm everyone's understanding aligns
- Plan next steps and validation activities

📊 Canvas Completion Checklist

👥 User Needs Section:

- [ ] 3-5 specific user problems clearly articulated
- [ ] Problems are ranked by priority/importance
- [ ] Each problem is tied to specific user types or workflows
- [ ] Problems are actionable (not too broad or vague)

🎯 Solution Concept Section:

- [ ] High-level approach is clear and understandable
- [ ] Concept directly addresses the prioritized user needs
- [ ] Approach is feasible given known constraints
- [ ] All participants understand and agree with direction

⚙️ Functional Requirements Section:

- ☐ Core functions are specific and testable
- ☐ Data inputs and outputs are clearly defined
- ☐ Integration requirements are technically feasible
- ☐ Requirements are prioritized (must-have vs nice-to-have)

🖥️ Interface Requirements Section:

- ☐ User workflows are step-by-step clear
- ☐ Output formats match user needs and existing systems
- ☐ Interface considerations include accessibility needs
- ☐ Interaction patterns are familiar to users

📋 Technical Constraints Section:

- ☐ Infrastructure limitations are documented
- ☐ Security and compliance requirements are clear
- ☐ Resource constraints (time, budget, staff) are realistic
- ☐ Dependencies on other systems are identified

📊 Success Criteria Section:

- ☐ Metrics are specific and measurable
- ☐ Success criteria are achievable and time-bound
- ☐ Both technical and user satisfaction metrics included
- ☐ Criteria can be validated through testing

🔧 Troubleshooting & Adaptations

🤔 "Users and developers are speaking different languages"

Common Issue: Technical jargon vs user terminology creating confusion

Solutions:

- **Assign a translator**: Someone who understands both sides
- **Use visual aids**: Sketches, mockups, existing system screenshots
- **Define terms**: Create shared vocabulary list during the session
- **Focus on "what" before "how"**: Understand functionality before implementation

Facilitation Tips:

- "Let me restate that in different terms..."
- "Can you show us an example of what you mean?"
- "What would that look like from your perspective?"

📈 "Requirements keep expanding during the session"

This is Normal but Needs Management:

Prevention Strategies:

- **Set scope boundaries clearly at start**: "Today we're focusing on core functionality"
- **Use a parking lot**: Capture good ideas that are out of scope
- **Time-box discussions**: "Let's spend 10 minutes on this topic"

When Scope Creeps:

- **Acknowledge the value**: "That's a great idea..."
- **Prioritize ruthlessly**: "Is this more important than what we already identified?"
- **Plan for later phases**: "Let's capture this for version 2"

⚖️ "Users disagree on requirements"

Different User Types = Different Needs (Often!)

Strategies:

- **Understand the root need**: Why do they want different things?
- **Look for flexible solutions**: Can the system serve both needs?
- **Prioritize by impact**: Which users are most critical to project success?
- **Plan phased approach**: Address highest priority needs first

Example Resolution:

- Field users want mobile interface, office users want desktop
- Solution: Responsive design that works on both
- Or: Start with mobile (higher priority), add desktop later

📋 Output Templates & Validation

Requirements Documentation Template

```
# [Solution Name] Requirements Definition
Date: [Workshop Date] | Participants: [List names and roles]

## 👥 Prioritized User Needs
1. [Highest priority need - specific problem statement]
2. [Second priority need - specific problem statement]
3. [Third priority need - specific problem statement]

## 🎯 Solution Concept
[2-3 sentences describing the high-level approach]

## ⚙️ Functional Requirements
### Must Have:
- [ ] [Specific function the system must perform]
- [ ] [Another critical function]

### Should Have:
- [ ] [Important but not critical function]

### Could Have:
```

```
- [ ] [Nice-to-have function for future consideration]

## 🖥️ Interface Requirements
### User Workflows:
1. [Step-by-step user process]
2. [Alternative workflow if needed]

### Output Formats:
- [What users receive and in what format]
- [Integration requirements with existing systems]

## 📋 Technical Constraints
- [Infrastructure limitations]
- [Security/compliance requirements]
- [Resource constraints]

## 📊 Success Criteria
- [Measurable technical performance metric]
- [User satisfaction metric]
- [Adoption/usage metric]
```

Validation Activities (After Workshop)

Week 1: Internal Review

- [ ] Technical team reviews feasibility of all requirements
- [ ] Project manager reviews scope and resource implications
- [ ] Stakeholder lead reviews user representation and priorities

Week 2: User Validation

- [ ] Send requirements summary to workshop participants for confirmation
- [ ] Share with 2-3 additional users who weren't in workshop
- [ ] Ask specific validation questions (see below)

Week 3: Final Documentation

- [ ] Incorporate feedback and finalize requirements
- [ ] Create technical specifications document
- [ ] Plan development approach and timeline

User Validation Questions

- Does this accurately represent your most important needs?
- Are we missing any critical functionality?
- Do the success criteria match how you'll evaluate the solution?
- Are the proposed interfaces compatible with your workflows?
- What concerns do you have about this approach?

🔄 Related Tools & Next Steps

Before This Tool:

- **Stakeholder Mapping Workshop** - Identify who should participate
- **Discovery Interview Blueprint** - Understand user needs in depth

After This Tool:

- **User Testing Protocol** - Test solutions against these requirements
- **Technical Validation Checklist** - Verify implementation meets requirements

Alternative Approaches:

- **User Journey Mapping Kit** - If workflows are very complex
- **Output Validation Checklist** - If requirements are mostly about data formats

---

📚 Source Attribution

Primary Methodology Sources:

- **Meeting Notes - Technical Development CoDesign Toolkit Working Group** - Requirements co-specification approach
- **Solution Co-Development Toolkit Narrative** - User-centered requirements definition
- **NSITE Solution Project Requirements and Expectations** - Collaborative specification processes

Canvas Design Inspiration:

- Business Model Canvas methodology adapted for technical requirements
- Lean UX and Design Thinking collaborative workshop techniques
- Agile user story and acceptance criteria frameworks

💬 Community Discussion

Share your experience:

- What was the most challenging part of facilitating requirements definition?
- How did you handle disagreements between users and technical constraints?
- What adaptations did you make for remote/distributed teams?
- What requirements did you miss that came up later in development?

Tool improvements:

- What sections would you add or modify?
- How do you handle very complex or technical domains?
- What validation methods work best for your context?

🔧 **Tool Maintainer:** @your-username | 📅 **Last Updated:** [Today's Date] | 📌 **Version:** 1.0