Change Management in the Age of AI - Companion Repository

This repository serves as a companion site to the book **Change Management in the Age of AI** — a comprehensive guide for leaders navigating complex organizational and individual transformation in an era of exponential technological change.

The book is available on Amazon

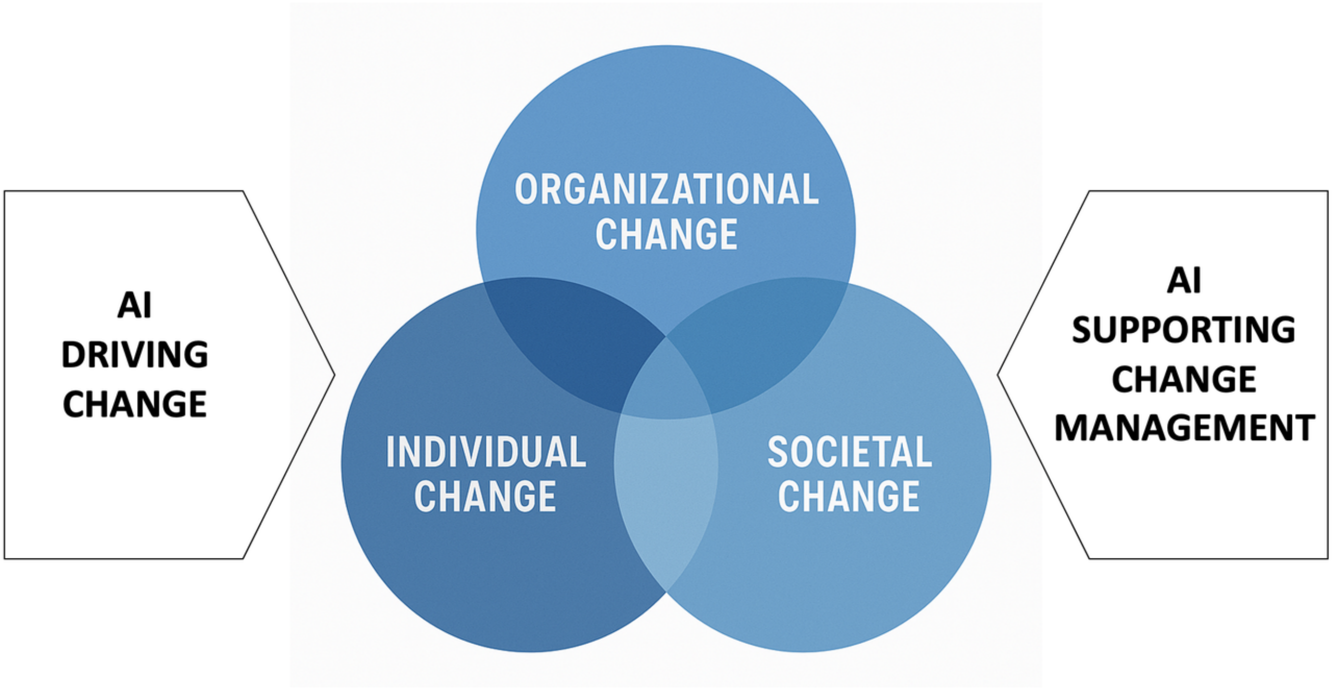
About the Book

Despite decades of change management research and practice, failure rates remain stubbornly high — with up to 88% of business transformations failing to achieve their original ambitions. This persistence of failure signals that traditional approaches, which assume gradual and predictable transformation, are fundamentally inadequate for today's grand challenges. Among these challenges, AI stands out as particularly transformative, reshaping how we work, compete, and create value at a fundamental level while amplifying all other challenges.

The book addresses the "exponential gap": the widening chasm between how fast AI capabilities evolve and how slowly our organizations, mindsets, planning processes, and governance structures adapt. Success requires transforming how we think about change itself — moving from discrete projects with clear endpoints to continuous adaptation as an individual and collective capability.

Key Themes: AI Tools for Complex Change Management

AI as Both Disruptor and Enabler: AI plays a dual role throughout the book — as the source of disruption requiring new forms of leadership, and as a tool that, properly deployed, enables that leadership to succeed. The task isn't choosing between these possibilities but holding them in productive tension — leveraging AI's capabilities while constraining its risks, using it to amplify human capability while ensuring humans remain genuinely in control.



Individual Change: Effective change management starts with individual transformation. The primary constraint in navigating complex change isn't technical knowledge but rather the individual's capacity to process complexity, tolerate ambiguity, and continuously reconstruct their mental models. The book draws on developmental psychology to distinguish three levels of adult consciousness, with the AI age increasingly demanding a "self-transforming mind" that can hold multiple contradictory perspectives simultaneously and view identity itself as fluid rather than fixed. AI serves as a developmental tool itself — AI coaches can provide 24/7 support for reflection, offer Socratic questioning, and help leaders test assumptions, democratizing access to developmental support.

Organizational Change: Resistance to change manifests across three interconnected contexts: (1) **Formal context** — structures, processes, metrics, and incentive systems; (2) **Social context** — trust levels, communication patterns, and psychological contracts; and (3) **Mental context** — collective mindsets and paradigms. Organizations can use AI itself to manage change: sentiment analysis reveals hidden concerns, predictive analytics identify adoption barriers before they become crises, and AI agents can provide personalized support at scale. The book emphasizes creating "hybrid intelligence" rather than replacement automation — maintaining human judgment while leveraging AI's analytical power.

Societal Change: Organizational success ultimately depends on healthy societal ecosystems. The book draws on complexity theory to explain why societal transitions are often fundamentally nonlinear and path-dependent, crisis-driven rather than gradually managed. Building societal resilience requires accepting some inefficiency to maintain redundancy, cultivating diversity, ensuring decision-makers bear consequences, and maintaining optionality rather than premature optimization. Leadership in the AI age is unavoidably political — building coalitions, negotiating between competing interests, and forging collective action despite fragmentation.

Practical Tools and Frameworks: Throughout, the book emphasizes practical application. Each chapter concludes with actionable tools and frameworks that leaders can implement immediately, including: Immunity to Change mapping, double-loop learning, personal mastery frameworks, organizational network mapping, behavioral tracking, scenario planning, agent-based modeling, and participatory AI governance.

The book's ultimate message is one of tempered but genuine optimism. The challenges are real and urgent, but human institutions have navigated previous transformations that seemed overwhelming at the time. Success requires abandoning the comfortable fiction that we can predict and plan our way through this transition, and instead building the capacity for continuous learning and adaptation.

About This Repository

This repository contains practical AI agent examples and tools for change management. The main components are:

- **`AgentExamples/`**: Jupyter notebooks demonstrating AI use cases in change management
- **`QUICK_START.md`**: Quick start guide for getting started with the notebooks
- **`requirements.txt`**: Python dependencies for the project
- **`static/`**: Static assets including the overview diagram

The Jupyter notebooks in the `AgentExamples/` directory are **practical illustrations** of AI use cases in change management at both the **individual and organizational levels**. These notebooks demonstrate how AI agents can be deployed to support the frameworks and tools described in the book:

- **Individual-Level Tools**: Notebooks like `psychological_safety_coach.ipynb` and `socratic_partner.ipynb` illustrate how AI can serve as a developmental tool for personal growth, providing coaching, Socratic questioning, and reflection support that helps individuals expand their capacity for complexity and ambiguity. Notebooks like `interestingness_evaluator_agent.ipynb` and `left-right-hemisphere-analyzer.ipynb` show how AI can support complex analysis, idea evaluation, and multi-perspective thinking that enables better decision-making in uncertain environments. The `immunitytochange.ipynb` notebook implements Robert Kegan's Immunity to Change framework to help individuals identify hidden barriers and competing commitments that prevent meaningful change. The `contrarian_intelligence.ipynb` notebook challenges conventional wisdom and identifies hidden assumptions, helping individuals and organizations surface alternative perspectives on complex topics.

- **Organizational-Level Tools**: Notebooks like `multi_agent_conversation.ipynb` and `focus_group_agent.ipynb` demonstrate how AI can facilitate organizational decision-making, simulate stakeholder perspectives, and help diagnose resistance across formal, social, and mental contexts. Notebooks like `analogy_finder_agent.ipynb` illustrate how AI can help generate cross-disciplinary insights and reframe problems, supporting the kind of creative thinking needed to navigate complex change. The `contrarian_intelligence.ipynb` notebook challenges conventional wisdom and identifies hidden assumptions underlying organizational narratives, helping surface alternative perspectives that can reveal blind spots in strategic thinking. The `resistance_to_change_analyzer.ipynb` notebook analyzes user adoption patterns and resistance to change initiatives, providing data-driven insights and actionable recommendations for different user segments. The `social_physics_analyzer.ipynb` notebook examines how information flows through social networks and how collective intelligence emerges from individual interactions, using network analysis and social physics principles.

- **Societal-Level Tools**: The notebook `dilemma_resolution.ipynb` uses LangChain's deep agent framework to implement Charles Hampden-Turner's Dilemma Surfacing techniques for resolving complex organizational or societal dilemmas through multi-stakeholder simulation. The notebook `scenario_planning.ipynb` provides a comprehensive scenario planning exercise for analyzing societal-level challenges, using established frameworks like Shell's Scenario Planning Method and STEEP analysis to systematically explore future uncertainties and generate actionable scenarios. The `survey_simulation.ipynb` notebook demonstrates how to simulate social science experiments and surveys using large language models, allowing you to simulate survey responses across different segments (countries, demographics, time periods) and compare results with reference survey data.

Each notebook is a working example that you can run, modify, and adapt for your own change management challenges. They use LangChain's deep agent framework to build sophisticated AI agents that go beyond simple chatbots to provide structured, multi-step reasoning and analysis.

Agent Examples - LangChain Deep Agent Framework

This directory contains AI agent examples using LangChain's deep agent framework. Each notebook demonstrates how to build sophisticated AI agents using LangGraph and helper functions. Many notebooks include interactive Gradio interfaces for easy interaction.

Structure

```
AgentExamples/
├── README.md                          # This file
├── requirements.txt                    # Python dependencies
├── activate.sh                         # Virtual environment activation
script
├── helpers/                            # Helper functions and utilities
│   ├── __init__.py
│   ├── agent_helpers.py               # Core agent helper functions
│   ├── openrouter_client.py           # OpenRouter API client
│   ├── input_processing.py             # Text, PDF, and image processing
│   ├── interestingness_visualization.py  # Visualization for
interestingness evaluator
│   ├── markdown_helpers.py            # Markdown processing utilities
│   ├── pdf_generator.py               # PDF report generation
│   ├── pdf_helpers.py                 # PDF utilities with image support
│   ├── report_helpers.py             # Report formatting utilities
│   ├── chat_pdf_helpers.py            # Chat PDF generation utilities
│   ├── gradio_helpers.py              # Gradio server management
utilities
│   ├── state_helpers.py               # Agent state display utilities
│   └── visualization.py               # Graph and message visualization
├── reports/                            # Generated PDF reports
├── analogy_finder_agent.ipynb         # Cross-disciplinary analogy finder
├── contrarian_intelligence.ipynb      # Contrarian intelligence analysis
├── dilemma_resolution.ipynb           # Multi-stakeholder dilemma
resolution
├── focus_group_agent.ipynb            # Focus group simulation with AI
personas
├── immunitytochange.ipynb             # Immunity to Change mapping
framework
├── interestingness_evaluator_agent.ipynb  # Multi-dimensional idea
```

```
evaluator
├── left-right-hemisphere-analyzer.ipynb  # Dual-hemisphere cognitive
analysis
├── multi_agent_conversation.ipynb     # Multi-agent organizational
decision-making
├── psychological_safety_coach.ipynb   # Psychological safety coaching
assistant
├── resistance_to_change_analyzer.ipynb  # User resistance and adoption
pattern analysis
├── scenario_planning.ipynb            # Societal scenario planning
exercise
├── social_physics_analyzer.ipynb      # Social network and information
flow analysis
├── socratic_partner.ipynb             # Socratic teaching and learning
partner
└── survey_simulation.ipynb            # Social science survey simulation
tool
```

Helper Functions

The `helpers/` directory provides reusable utilities for building agents:

`agent_helpers.py`

Core agent framework utilities:

- `create_openrouter_llm()` - Create LangChain LLM with OpenRouter API
- `display_agent_graph()` - Visualize agent graphs in Jupyter
- `format_messages()` - Format message history for display
- `render_agent_output()` - Render agent output with proper formatting
- `create_agent_state()` - Create TypedDict for agent state

`openrouter_client.py`

Unified OpenRouter API client for direct API access:

- `OpenRouterClient` - Client class for interacting with OpenRouter API
- `get_completion()` - Get text completions from various LLM models
- `get_completion_stream()` - Get streaming completions
- Supports all OpenRouter models (GPT, Claude, Gemini, etc.)
- Handles authentication and error messages automatically

`input_processing.py`

Input handling for various formats:

- `pdf_to_text()` - Extract text from PDF files
- `image_to_text()` - Extract text from images using vision models
- `image_url_to_text()` - Extract text from images via URL
- `process_input()` - Unified input processing for text, PDF, and images
- `process_input_with_images()` - Process input with image support

- `image_file_to_base64()` - Convert image files to base64
- `image_url_to_base64()` - Convert image URLs to base64
- Supports multiple input types: text, PDF files, image files, and image URLs

`pdf_generator.py`

PDF report generation:

- `generate_pdf_report()` - Create PDF reports from conversation history
- `format_conversation_for_pdf()` - Format messages for PDF export
- Support for markdown formatting and styling

`visualization.py`

Graph and data visualization:

- `plot_agent_graph()` - Plot agent graphs with matplotlib/networkx
- `display_message_history()` - Display formatted message history
- `create_interactive_graph()` - Create interactive graph visualizations

`interestingness_visualization.py`

Specialized visualizations for interestingness evaluator:

- Spider charts for multi-dimensional scoring
- Score comparison visualizations

`markdown_helpers.py` and `report_helpers.py`

Formatting and report utilities for structured output generation:

- `process_inline_formatting()` - Process inline markdown formatting
- `markdown_to_html()` - Convert markdown to HTML
- `parse_json_to_md()` - Parse JSON to markdown
- `generate_html_report()` - Generate HTML reports from evaluation results

`pdf_helpers.py` and `chat_pdf_helpers.py`

PDF generation utilities with image support:

- `markdown_to_pdf_with_images()` - Convert markdown to PDF with embedded images
- `add_image_to_markdown()` - Add images to markdown content
- `generate_chat_pdf_html()` - Generate HTML for chat-based PDF reports

`gradio_helpers.py`

Gradio server management utilities:

- `stop_gradio_servers()` - Stop all running Gradio servers or servers on a specific port
- `ensure_gradio_port_free()` - Ensure a specific port is free for Gradio
- Prevents "Cannot find empty port" errors when running multiple notebooks with Gradio interfaces
- Cross-platform support (Windows, macOS, Linux)

`state_helpers.py`

Generic agent state display utilities:

- `display_agent_state()` - Generic function to display agent graph state in a formatted, readable way
- Replaces notebook-specific state display functions
- Supports custom field formatters and formatting options
- Handles nested structures, JSON strings, and long content with truncation
- Displays timestamps, model names, conversation history, and results

Usage

Each notebook is self-contained and can be run independently:

1. **Setup**: Import helper functions and configure the LLM

```
import sys
from pathlib import Path

# Add AgentExamples directory to path to import helpers
agent_examples_dir = Path.cwd()
if str(agent_examples_dir) not in sys.path:
    sys.path.insert(0, str(agent_examples_dir))

from helpers import (
    create_openrouter_llm,
    display_agent_graph,
    # ... other helpers
)
```

2. **Define State**: Create TypedDict for agent state management

3. **Define Nodes**: Create functions for each agent node

4. **Build Graph**: Use StateGraph to connect nodes

5. **Run**: Invoke the compiled graph with initial state

Using OpenRouterClient Directly

If you need direct access to the OpenRouter API (outside of LangChain):

```
from helpers import OpenRouterClient

client = OpenRouterClient()
response = client.get_completion(
    prompt="Your question here",
    model="openai/gpt-5-nano",
    temperature=0.7
)
```

Examples

1. Analogy Finder (`analogy_finder_agent.ipynb`)

Finds cross-disciplinary analogies for complex concepts using a multi-step agent workflow:

- Extracts core notions from input text
- Generates analogies from diverse sectors (biology, physics, history, etc.)
- Formats output with mappings and insights
- PDF report generation

2. Dilemma Resolution (`dilemma_resolution.ipynb`)

Uses LangChain's deep agent framework to implement Charles Hampden-Turner's Dilemma Surfacing techniques:

- Multi-stakeholder simulation for resolving complex organizational or societal dilemmas
- Three specialized agents: Sustainability Advocate, Financial Analyst, Regulatory Expert
- Each agent generates proposals that are critiqued by others
- Synthesizer agent integrates perspectives into actionable solutions
- Interactive Gradio interface
- PDF report generation

3. Focus Group Agent (`focus_group_agent.ipynb`)

Simulates realistic focus group discussions using AI agents with diverse personas:

- Generates representative focus group members based on topic
- Customizable participant personas (demographic, psychographic, communication style)
- Professional moderator agent facilitates discussion
- Synthesizer agent creates summary and insights
- Interactive Gradio interface
- PDF report generation

4. Interestingness Evaluator (`interestingness_evaluator_agent.ipynb`)

Evaluates ideas across 7 dimensions using parallel evaluation nodes and structured output:

- Novelty and Originality
- Explanatory and Predictive Power
- Testability (Falsifiability)
- Clarity and Precision
- Coherence and Logical Consistency
- Boldness and Heuristic Value
- Elegance and Parsimony
- Includes spider chart visualization and counterarguments

- Interactive Gradio interface with multiple input options
- PDF report generation with image rendering
- Supports text, PDF files, image files, and image URLs

5. Left-Right Hemisphere Analyzer (`left-right-hemisphere-analyzer.ipynb`)

Analyzes text, images, and PDFs using a dual-hemisphere cognitive processing approach:

- **Left-Hemisphere Analyst**: Analytical, factual, and structured analysis focusing on details, logic, and evidence
- **Right-Hemisphere Interpreter**: Holistic, intuitive interpretation focusing on big-picture meaning, context, and creative connections
- **Synthesizer**: Integrates both perspectives into a deeper overall understanding
- Interactive Gradio interface with multiple input options
- Supports text, PDF files, image files, and image URLs
- PDF report generation with embedded images
- Model selection for all three agents

6. Multi-Agent Conversation (`multi_agent_conversation.ipynb`)

Demonstrates multi-agent systems for organizational decision-making:

- Multiple agents with distinct personas
- Structured conversation flow
- Automatic summarization
- PDF report generation

7. Psychological Safety Coach (`psychological_safety_coach.ipynb`)

AI coaching assistant for psychological safety in difficult conversations and change management:

- Role-play scenarios for difficult conversations
- Guidance on handling change and conflict
- Interactive Gradio chat interface
- Conversation export to PDF
- Model selection (default: `openai/gpt-5-nano`)

8. Scenario Planning (`scenario_planning.ipynb`)

Comprehensive scenario planning exercise for analyzing societal-level challenges using established frameworks:

- **Topic Definition**: Clearly articulate the societal challenge or question with time horizon and geographic scope
- **STEEP Analysis**: Systematically identify drivers of change across Social, Technological, Economic, Environmental, and Political categories
- **Uncertainty-Impact Matrix**: Map drivers by uncertainty and impact to identify key uncertainties
- **Tipping Points**: Detect critical thresholds and potential system shifts
- **Sensitivity Analysis**: Understand dependencies and vulnerabilities
- **Scenario Generation**: Create 2-4 distinct future scenarios based on key uncertainties

- **Implications Analysis**: Explore what each scenario means for stakeholders
- **Early Warning Signals**: Define indicators to monitor for scenario validation
- **Research-Enhanced Features**:
  - Scenario validation using research-based quality criteria
  - Cross-impact analysis to understand driver interdependencies
  - Research-based prompts incorporating best practices
- **Visualization**: Uncertainty-impact matrix visualization
- **PDF Report Generation**: Comprehensive PDF reports with all analyses and visualizations
- Based on Shell's Scenario Planning Method and established scenario planning frameworks
- Model selection (default: `openai/gpt-5-nano`)

9. Socratic Partner (`socratic_partner.ipynb`)

Socratic teaching and learning assistant that helps users master any topic through guided discovery:

- Guided discovery through targeted questions
- Adaptive pacing based on learner's understanding
- Vivid analogies and metaphors for clarity
- Active dialogue to surface and resolve confusion
- Building understanding layer by layer
- Real-world relevance and application
- Interactive Gradio chat interface
- Conversation export to PDF
- Model selection (default: `openai/gpt-5-nano`)

10. Immunity to Change (`immunitytochange.ipynb`)

Implements Robert Kegan and Lisa Laskow Lahey's Immunity to Change framework to help individuals and organizations identify hidden barriers to change:

- Maps current behaviors that work against improvement goals
- Identifies hidden competing commitments and underlying fears
- Surfaces big assumptions that maintain the status quo
- Suggests specific behavioral goals to overcome resistance
- Generates comprehensive PDF reports with structured analysis
- Based on developmental psychology principles
- Helps understand why change efforts fail despite good intentions

11. Resistance to Change Analyzer (`resistance_to_change_analyzer.ipynb`)

Analyzes user resistance and adoption patterns for change initiatives using data-driven insights:

- Imports and analyzes user data from change initiatives
- Categorizes users into segments: Champions, Early Adopters, Pragmatists, Skeptics, Resisters
- Visualizes distribution of user adoption patterns
- Generates specific, actionable recommendations for each segment
- Uses AI-powered segmentation to identify patterns
- Applies change management frameworks to understand resistance
- Provides prioritized interventions with timelines

- Supports export of segmented data and recommendations

12. Social Physics Analyzer (`social_physics_analyzer.ipynb`)

Examines how information flows through social networks and how collective intelligence emerges:

- Analyzes network structure, centrality measures, and community detection
- Creates force-directed graph visualizations of social networks
- Uses network analysis and graph theory to understand social structure
- Applies social physics principles to understand information flow and influence patterns
- AI-powered interpretation of network patterns and insights
- Generates comprehensive PDF reports with visualizations
- Works with email networks, collaboration networks, and other social structures

13. Survey Simulation (`survey_simulation.ipynb`)

Simulates social science experiments and surveys using large language models:

- Simulates survey responses across different segments (countries, demographics, time periods)
- Compares simulated results with reference survey data
- Supports multiple survey formats (scale ratings, distribution analysis)
- Statistical analysis with confidence intervals using scipy
- Visualizes results with bar charts and distribution plots with error bars
- Validates simulations against real survey data
- Based on research on predicting social science experiment results using LLMs
- Useful for testing hypotheses, exploring demographic differences, and understanding public opinion trends

14. Contrarian Intelligence (`contrarian_intelligence.ipynb`)

Challenges conventional wisdom and surfaces alternative perspectives on complex topics:

- Identifies dominant narratives and conventional wisdom from input text
- Uncovers hidden assumptions, implicit premises, and taken-for-granted beliefs
- Generates contrarian perspectives that challenge or invert assumptions
- Provides evidence, implications, and credibility assessments for each perspective
- Multi-step agent workflow: extract wisdom → identify assumptions → generate perspectives → format output
- PDF report generation with structured analysis
- Inspired by the Contrarian Intelligence Report framework
- Useful for strategic planning, decision-making, and identifying blind spots in organizational thinking

Key Features

- **Deep Agent Framework**: Uses LangGraph for complex agent workflows
- **OpenRouter Integration**: Unified access to multiple LLM models (default: `openai/gpt-5-nano`)
- **Model Selection**: Choose from multiple models including:
  - `openai/gpt-5-nano` (default)
  - `openai/gpt-5-mini`

    - `openai/gpt-5.2`
    - `anthropic/claude-haiku-4.5`
    - `google/gemini-3-flash-preview`
- **Interactive Interfaces**: Many notebooks include Gradio web interfaces for easy interaction
- **Multiple Input Types**: Support for text, PDF files, image files, and image URLs
- **Image Processing**: Automatic image description using vision models (GPT-4o)
- **PDF Export**: Generate PDF reports from conversations and analyses with embedded images
- **Image Rendering in PDFs**: Uploaded images and images from URLs are automatically included in PDF reports
- **Helper Functions**: Reusable utilities for common tasks
- **Gradio Server Management**: Automatic port management to prevent conflicts
- **State Display**: Generic state visualization for agent graphs
- **Jupyter-Optimized**: Designed for interactive notebook use
- **Clean Separation**: Application logic separated from framework code

Setup

Option 1: Use the Pre-configured Virtual Environment (Recommended)

A virtual environment has been created with all dependencies installed:

```
cd AgentExamples

# Activate the virtual environment
source venv/bin/activate  # On macOS/Linux
# or
source activate.sh       # Using the helper script

# Start Jupyter
jupyter notebook
# or
jupyter lab

# The virtual environment is registered as a Jupyter kernel
# Select "Python (ChangeManagementAI)" as the kernel when opening notebooks
```

Option 2: Create Your Own Virtual Environment

If you prefer to create your own virtual environment:

```
cd AgentExamples
python3 -m venv venv
source venv/bin/activate  # On macOS/Linux
pip install --upgrade pip
pip install -r requirements.txt
```

Requirements

The following packages are required (see `requirements.txt` for versions):

- **Core Framework**: langchain, langchain-core, langchain-openai, langgraph
- **Data Validation**: pydantic
- **Visualization**: matplotlib, networkx
- **Notebook Support**: ipython, jupyter
- **Environment**: python-dotenv
- **PDF Generation**: reportlab, weasyprint, markdown
- **Interactive Interfaces**: gradio
- **Input Processing**: pymupdf (PDF), pillow (images), requests (URL downloads)
- **Scientific Computing**: numpy, scipy
- **Data Analysis**: pandas, openpyxl (Excel support), seaborn (visualizations)

Note: For graph visualization, you may also need to install Graphviz system package:

- macOS: `brew install graphviz`
- Ubuntu/Debian: `sudo apt-get install graphviz`
- Windows: Download from Graphviz website

Environment Variables

Make sure you have your `OPENROUTER_API_KEY` set in your `.env` file in the repository root (not in AgentExamples). The `.env` file should be in the root directory of the repository:

```
OPENROUTER_API_KEY=sk-or-v1-your-key-here
```

All helper functions and the `OpenRouterClient` will automatically load this key from the environment.

Jupyter Kernel

The virtual environment has been registered as a Jupyter kernel named "Python (ChangeManagementAI)". When opening notebooks in Jupyter:

1. Select the kernel from the kernel dropdown menu (top right)
2. Choose "Python (ChangeManagementAI)" to use the pre-configured environment
3. This ensures all dependencies are available and the correct Python interpreter is used

If you need to re-register the kernel:

```
cd AgentExamples
venv/bin/python -m ipykernel install --user --name=changemanagementai --
display-name="Python (ChangeManagementAI)"
```

Usage Tips

Running Notebooks with Gradio Interfaces

Several notebooks (Psychological Safety Coach, Socratic Partner, Interestingness Evaluator, Left-Right Hemisphere Analyzer, Focus Group Agent, Dilemma Resolution) include Gradio web interfaces. To use them:

1. Run the notebook cells to start the Gradio interface
2. The interface will open in your browser automatically
3. Interact with the agent through the web interface
4. For chat-based interfaces: Type "end" in the chat to save the conversation to PDF and close the server
5. For analysis interfaces: The server automatically closes after completing an analysis

Input Options

Many notebooks support multiple input types with the following priority order:

1. **Text Input**: Direct text entry in the text box
2. **PDF Files**: Upload PDF documents (text will be extracted)
3. **Image Files**: Upload PNG or JPG images (content will be described using AI vision models)
4. **Image URLs**: Enter the URL of an image (content will be downloaded and described using AI vision models)

Note: Only one input type should be provided at a time. If multiple are provided, the priority order above determines which is used.

Model Selection

All notebooks support model selection. The default model is `openai/gpt-5-nano`, but you can choose from:

- `openai/gpt-5-nano` - Fast and cost-effective (default)
- `openai/gpt-5-mini` - Better quality, slightly more expensive
- `openai/gpt-5.2` - Latest model with enhanced capabilities
- `anthropic/claude-haiku-4.5` - Fast Claude model
- `google/gemini-3-flash-preview` - Google's latest preview model

PDF Reports

Many notebooks automatically generate PDF reports in the `reports/` directory. Reports include:

- Full conversation history or analysis results
- Formatted analysis and insights
- **Embedded images**: If an image was provided (via file upload or URL), it is automatically included in the PDF
- **Image URLs**: For images provided via URL, the URL is also displayed in the PDF
- Timestamps and metadata
- Visualizations (spider charts, graphs, etc.)

PDF reports are automatically generated after each analysis and saved in the `reports/` directory with timestamped filenames.