This is an early outline of the book *Troubleshooting Agile* by Jeffrey Fredrick and Douglas Squirrel. See the companion website http://troubleshootingagile.com for more about the book and related projects.

Introduction. We explain our goal for the book, the approaches we use to address organisational change, and suggest ways to use the book - for instance, starting with a relevant problem or mastering general techniques from Part I first.

Part I: Techniques for Change. In this part, we show the reader three generic ways to help her team change and improve. Each chapter starts with a short real-life story to show where the technique may be useful, followed by a description of the method, and finishes with one or more practical exercises the reader and her team can use to try out the method she's just learnt about. (In this outline we've just sketched the exercise for each section.)

Create Psychological Safety by Building Relationships. Exercise: Try coherence busting in each of the below scenarios, identifying at least five alternative explanations for each observation….

Obtain Valid Information with the Ladder of Inference. Exercise: Write a series of questions you might ask to find out how a colleague views a current situation. Next to each question, write the corresponding step of the ladder of inference.

Build Internal Commitment with Joint Design. Exercise: Pick a problem area where you have some responsibility and prepare your team to make a decision about it jointly. Fix a duration for your discussion, agree how you're going to decide by the end of that time, and encourage curiosity in the group as you discuss. Here are some example dialogues to inspire you...

Part II: Problems and Solutions. The chapters in this part cover specific situations agile teams encounter, and links each to a principle from the Agile Manifesto (see https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/). As in Part I, each chapter starts with a real-life story, goes on to an explanation of methods to use to address the problem, and finishes with practical exercises and actions the reader can try immediately. In this outline, we've just included the principles and a few of the exercises to give a flavour of the chapters.

We Take Forever to Ship. Principle: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

Our Customers Keep Changing Their Minds. Principle: "Welcome changing requirements, even late in development." Exercise: Ask three customers or customer representatives to explain their thinking about changing requirements. Use the Ladder of Inference to keep yourself curious and try to learn something new about their needs.

We Don't Know What Our Users Want. Principle: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

We Don't Have the Power to Remove Obstacles. Principle: "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done." Exercise: Write down three ways you can use coherence busting and other relationship-building methods from Part I to improve relations and trust between your team and outside managers.

We Never Get a Break to Fix Things. Principle: "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."

Our Technical Debt is Crippling Us. Principle: "Continuous attention to technical excellence and good design enhances agility."

We Don't Know How to Improve. Principle: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts

its behavior accordingly.” Exercise: Use your next retrospective to jointly inquire into a problem area and jointly design a solution, keeping the inquiry blameless.

Our Meetings Go On Forever. Principle: “The best architectures, requirements, and designs emerge from self-organizing teams.”

Our Estimates Are Hopelessly Wrong. Principle: “Working software is the primary measure of progress.”

We Work in Isolation. Principle: “Business people and developers must work together daily throughout the project.”

Part III: Putting It All Together. We provide three case studies drawn from our own experience in helping troubleshoot agile development teams. We illustrate how we applied the techniques in parts I and II to get the teams back on track.

Doing Everything Right - Except Delivering. The team has beautiful kanban cards, highly disciplined planning, and an immaculate set of coding practises and tests. There’s just one thing wrong - they don’t actually ship any code to customers. Giving them the psychological safety to question this state of affairs and change it allowed them to begin shipping valuable features that massively improved business results.

Disconnected From Deadlines. This development team drew up one list of their current projects, and another of the tasks that were urgently needed to meet external deadlines - and the two lists had no overlap at all. Jointly designing methods to address those external deadlines brought rapid success and a sense of accomplishment to the team.

The Sisyphus Team. The developers are in one room, marketing and sales in the other, and the two groups rarely talk. The devs are crunching through one task after another but have no sense of involvement or purpose. Helping them inquire into the reason for their work with the Ladder of Inference gave them a focus and the ability to participate in prioritising work for business value.