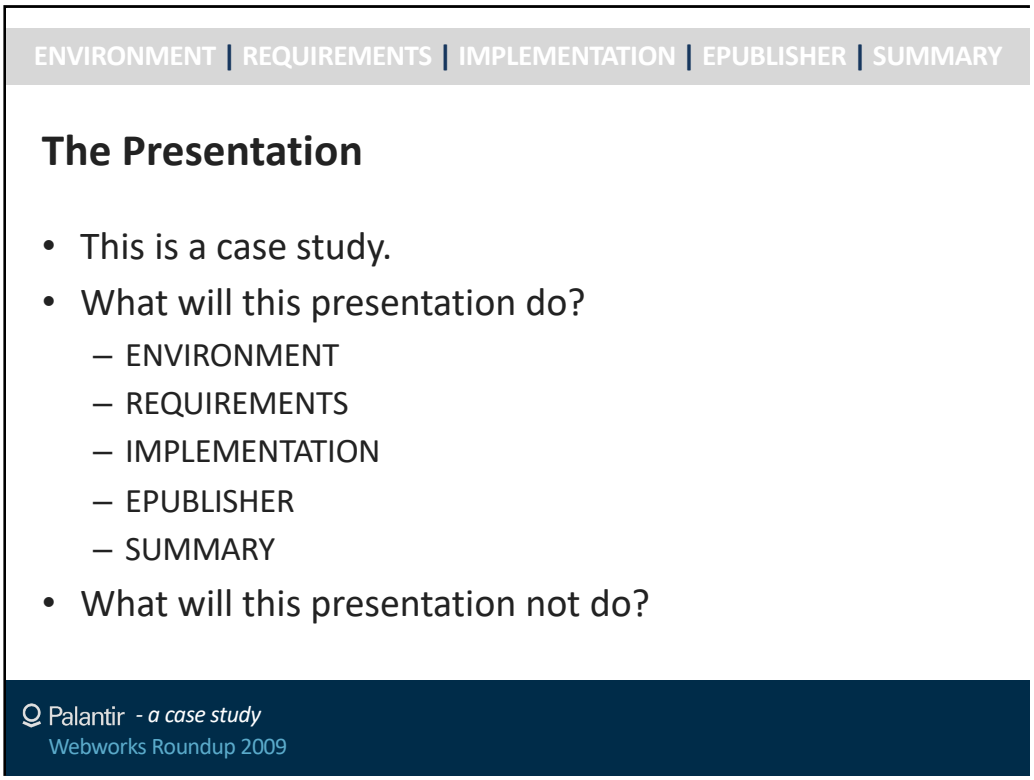
The title of my presentation is a bit outdated “1001 Nightly Builds.” As of this morning’s build report, Palantir has we have 4 main code trunk builds. The help release build, which goes out with our product, has over 1400 successful runs in this branch. 1436 to be exact.

In this simple statement, you can see some of the power of a build infrastructure. Our build infrastructure is integrated with the product, it is a repeatable structure, and it generates data that can be reported and analyzed.



A case study of how a documentation build infrastructure that uses ePublisher Software was developed in a small software company.

What it will do:
Explain the people, hardware, and software that make up a documentation build infrastructure

Walk through the build infrastructure at one firm

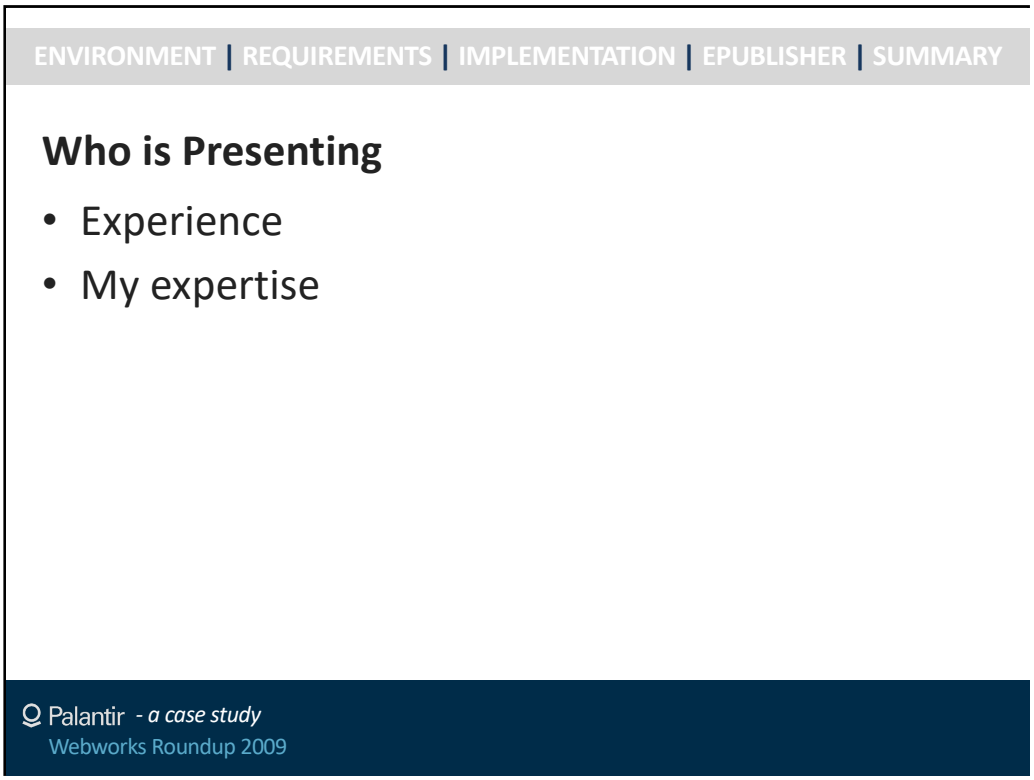
Describe the role ePublisher Software played in the build infrastructure at one firm.

Identify typical milestones and stumbling blocks in the development process and how to deal with them.

Generate ideas for others about how to develop an infrastructure.

What it won't do:
Teach you how to implement ePublisher Automap

Provide detailed explanations of tool configure or scripts

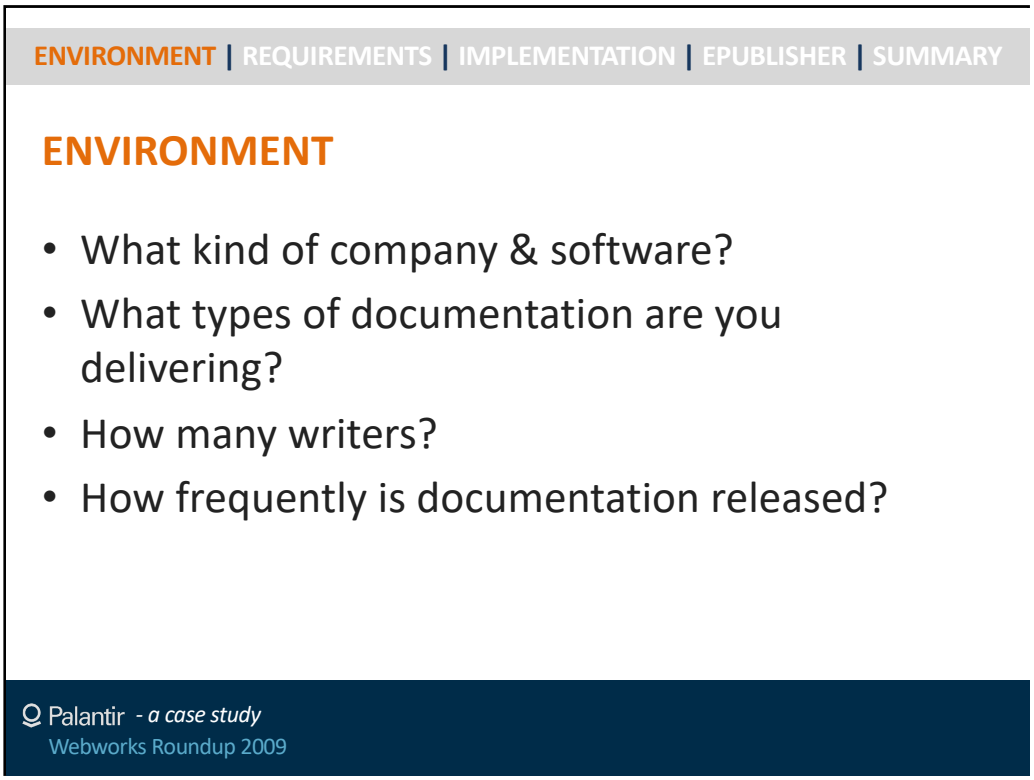


18+ years experience in the software industry, mostly in startups.

Working with Webworks on and off since 1998. This is the second build infrastructure I've put together.

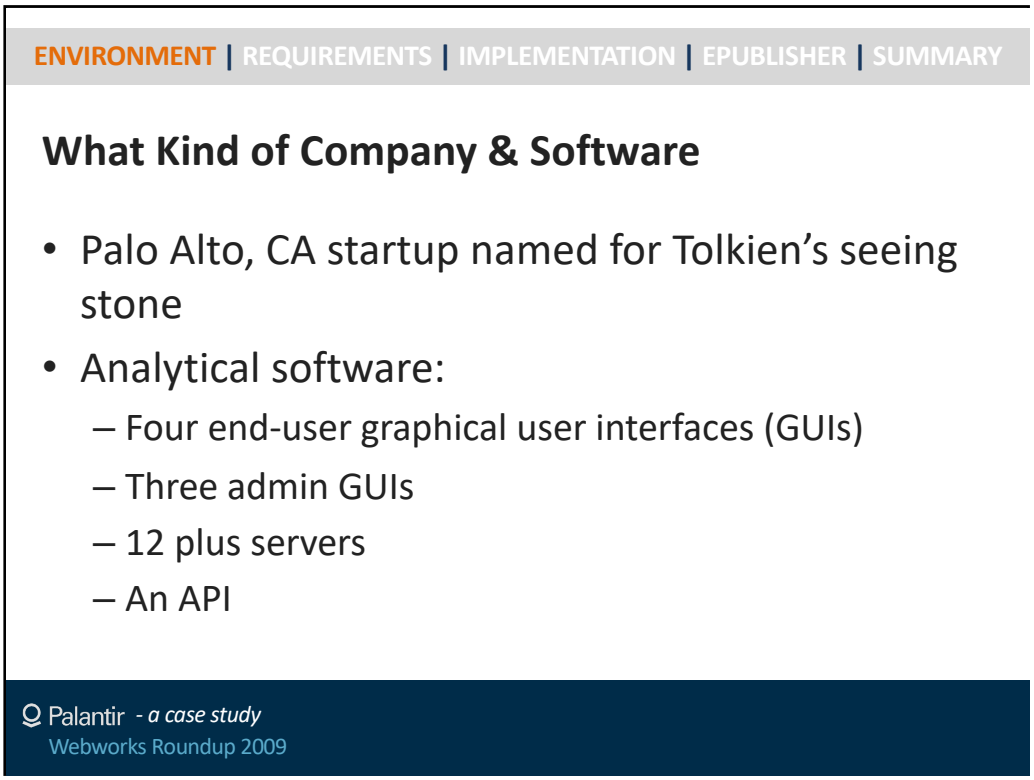
I am an expert in Palantir's build documentation build system. This is a subset of our larger build infrastructure.

I do not consider myself an expert in Framemaker, Webworks or even in any of the other tools I will talk about. However, I'll do my best to answer any questions that might arise during the presentation.



Every company has a documentation build infrastructure.

The environment at the company determines what kind of infrastructure is appropriate



Begun in 2004, 180 employees

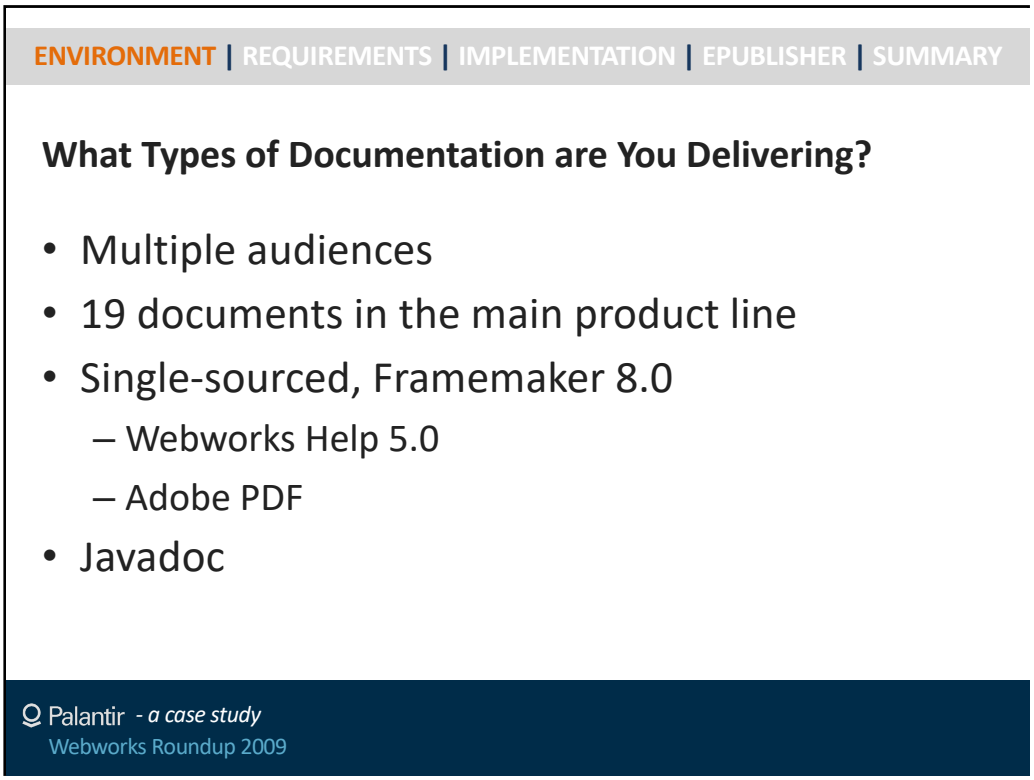
We are growing rapidly which figures into the stresses on documentation.

28 Engineers and 5 writers.

We use the agile development process. Approximately 4 major releases every year.

There are approximately 4 iterations per major releases. Documentation is attempting to release interim material with the iterations.

This means we release documentation externally 4 times a year and have 8 internal releases.



The audience for our documentation includes:

- Analysts using our Workspace and Web User clients
- Administrators using our Configuration Management and
- Integrators using our Dynamic Ontology Manager client (Java Swing client)
- Programmers using our APIs

WebWorks Help 5.0 installed through our product installers launched from Java Swing client (3 projects)

Web-based software (3 projects)

WebWorks Help 5.0 available from our Devzone Portal. This is a searchable build of all of our products. This single project contains 19 books.

PDF files available for download from our support portal

We also have a Javadoc for programmers to reference. This is its own build.

How Many Writers?

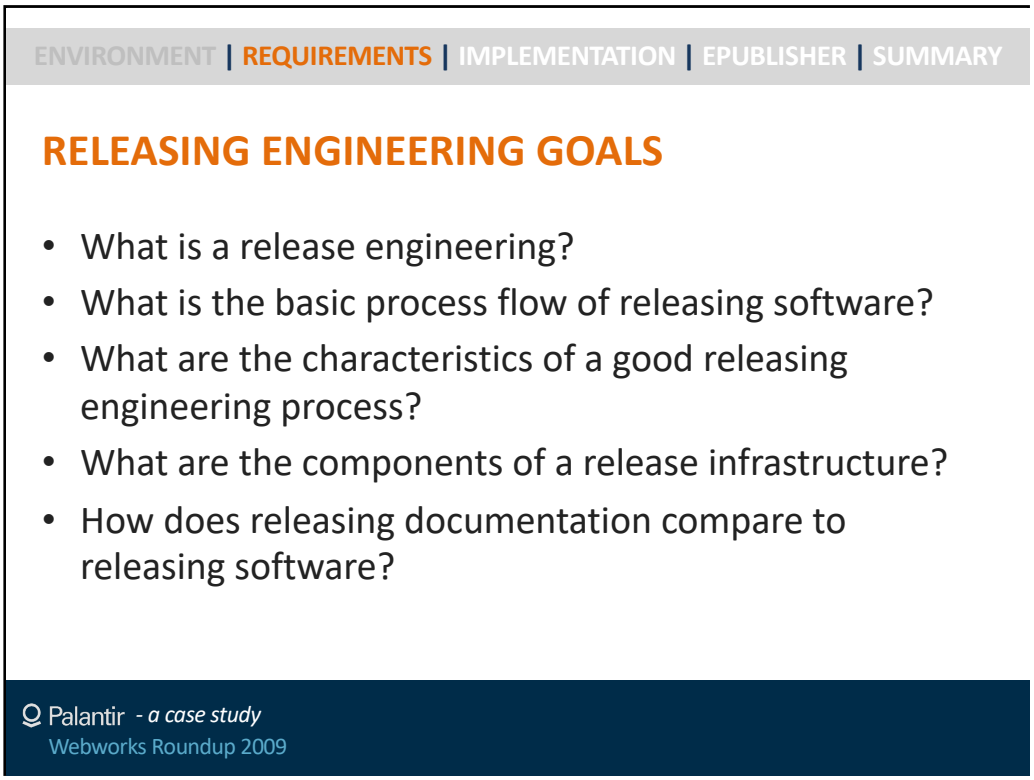
- 4 1/2 full time writers
- 1 part-time doc build engineer
- Average 15 years writing
- Average 6 years experience in Framemaker

How Frequently are You Releasing?

- 4 major releases a year
- Each release contains ~ 4 iterations
- Documentation team releases in tandem

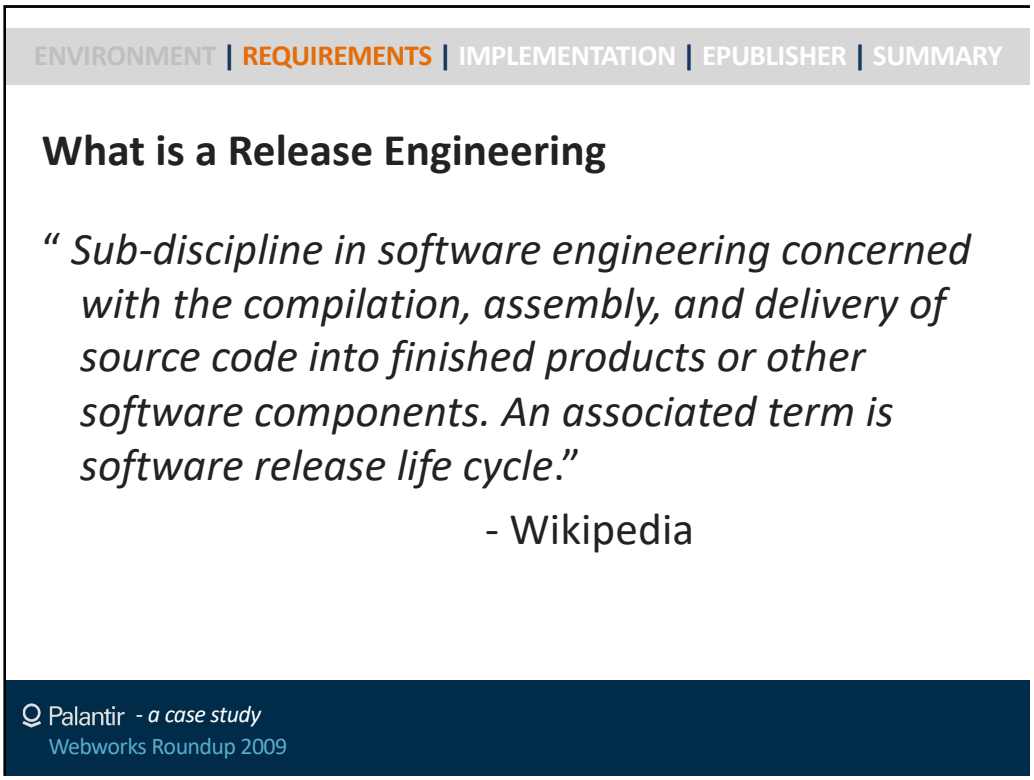
Environment Impacts Infrastructure

- What kind of documentation are you delivering?
- How integrated into the product is the help?
- How frequently are you delivering the documentation?
- How many writers producing how many documents?



It is important to understand where the documentation build fits into the software development process. This understanding can help you:

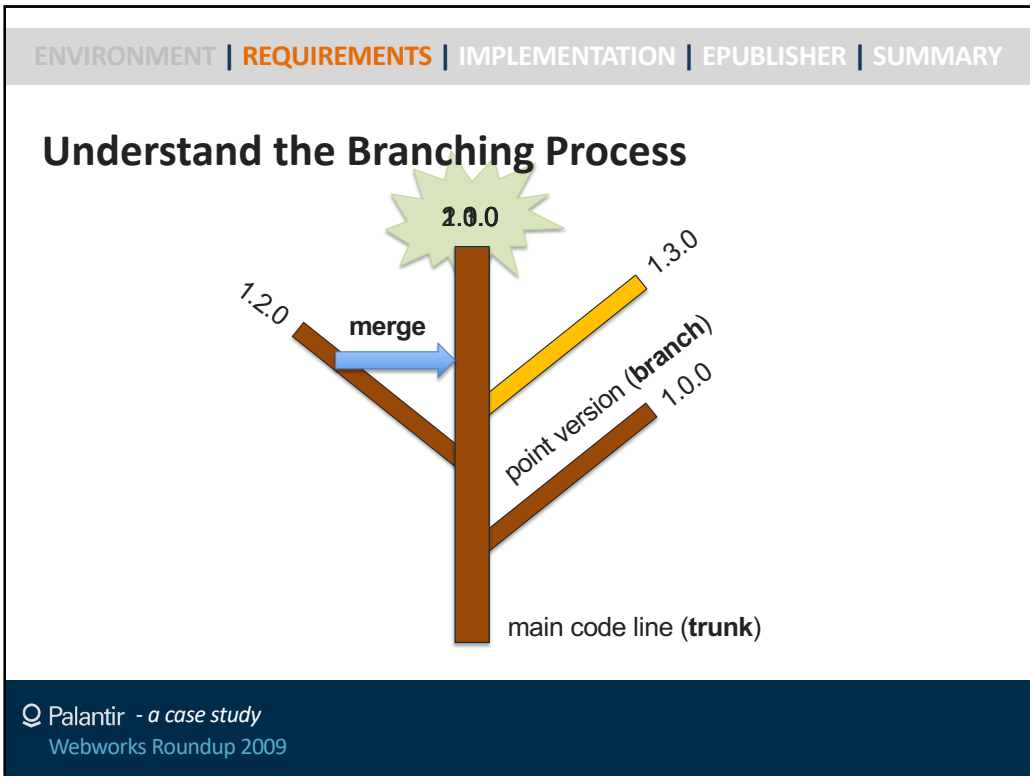
- Make an argument for creating a build structure if you don't have one
- Find key people in your organization who can move your project forward
- Identify the goals of a good documentation build infrastructure



Individual engineers "build" all or parts of a product to do unit testing

Release engineers build a version of a product for a release to customers

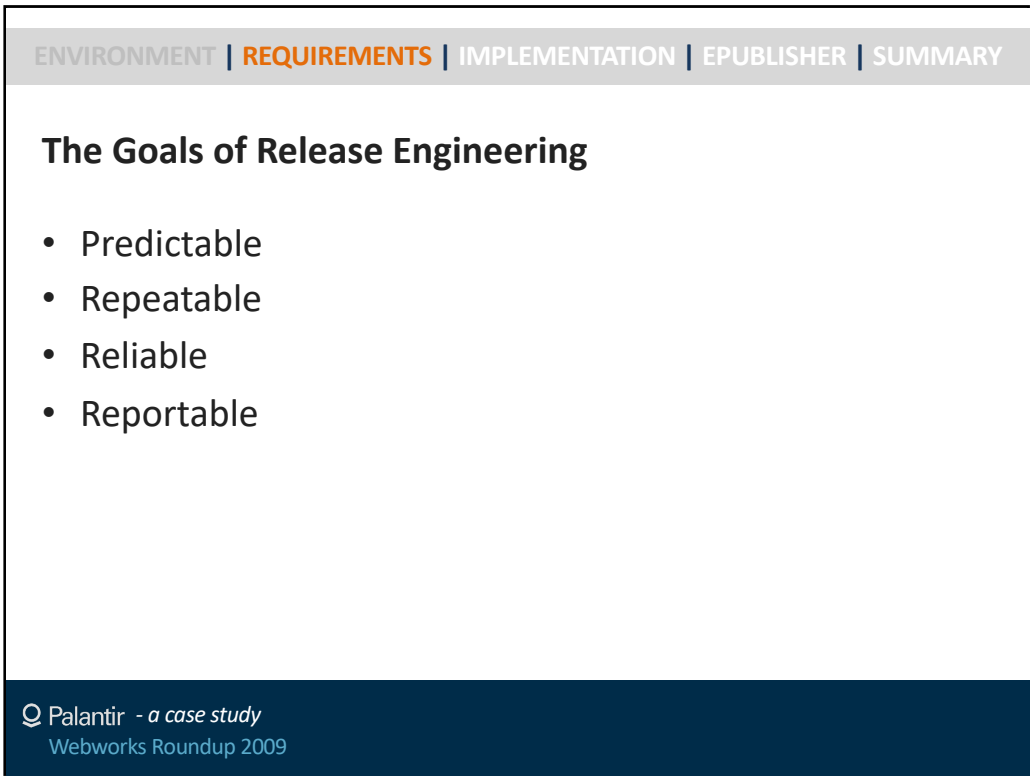
Typically a team in a company



Individual engineers “build” all or parts of a product to do unit testing

Release engineers build a version of a product for a release to customers

Typically a team in a company



How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

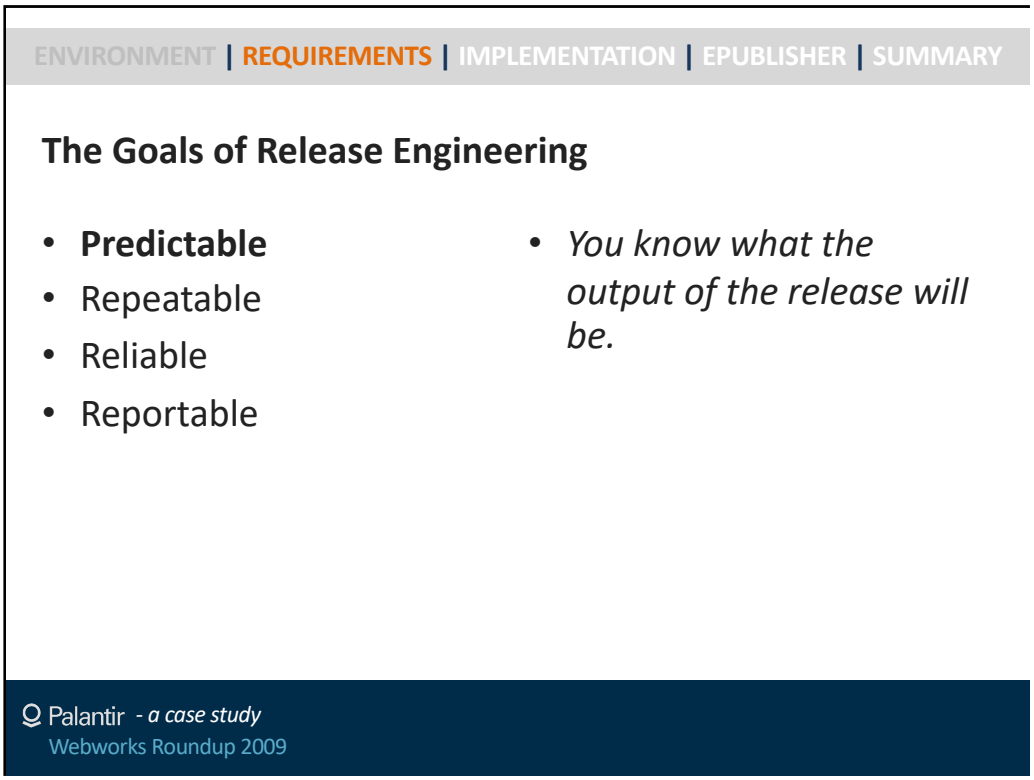
It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.



How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

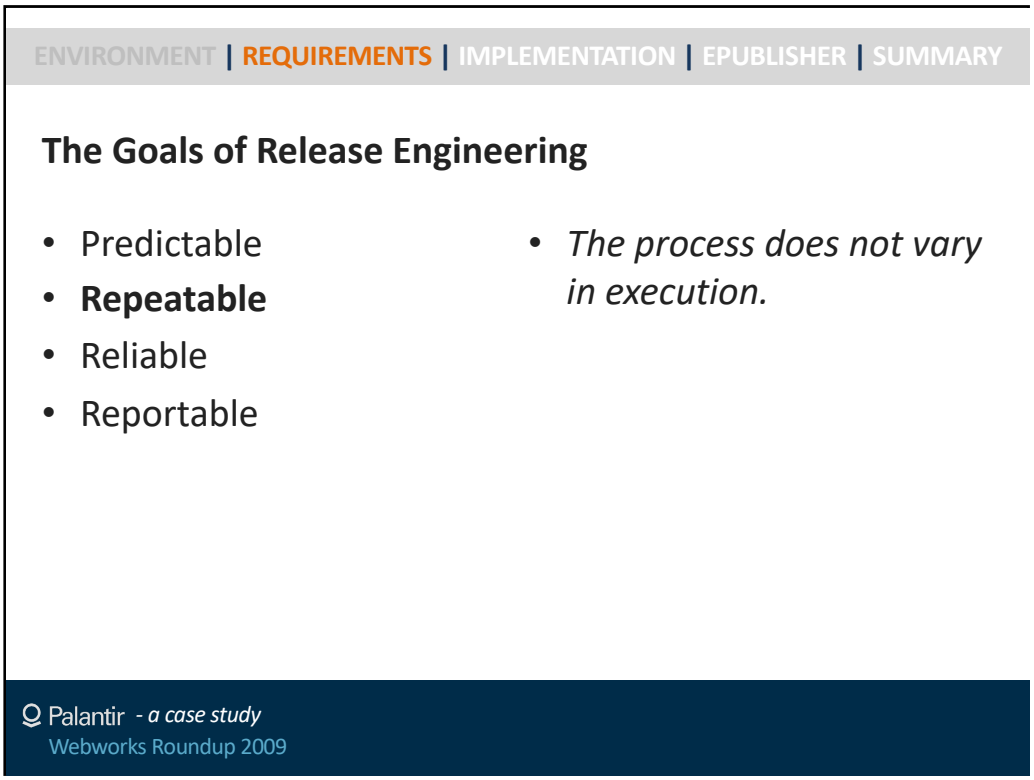
It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.



How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

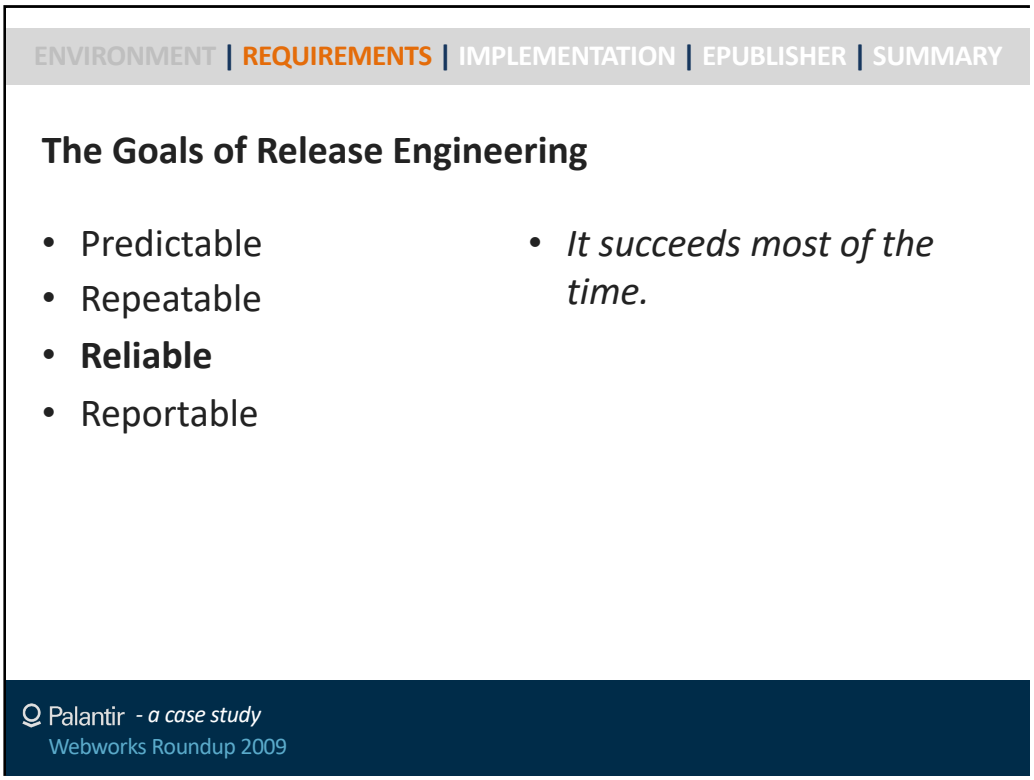
It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.



How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

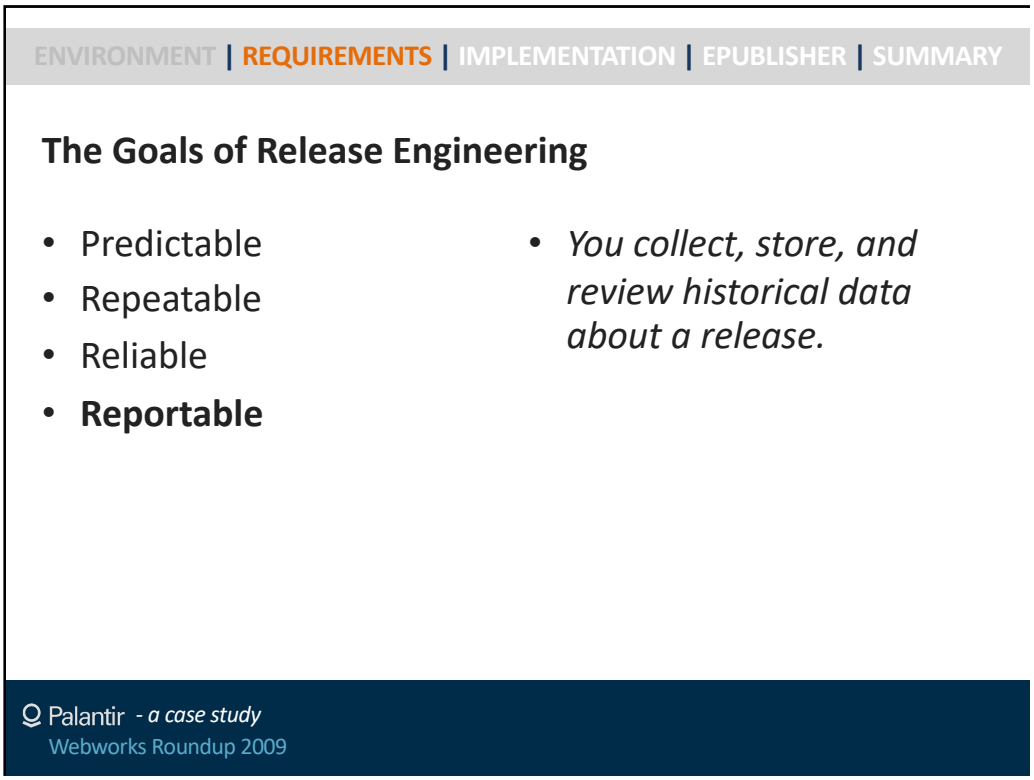
It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.



How do you know when you have “engineered” your documentation release.

Predictable

You know what you are going to get out of it. For example, we will get Webworks help and PDF.

This allows other portions of the product to know what is expected from your build.

Repeatable

The process runs in the same manner each time it runs. There is no variation.

Reliable

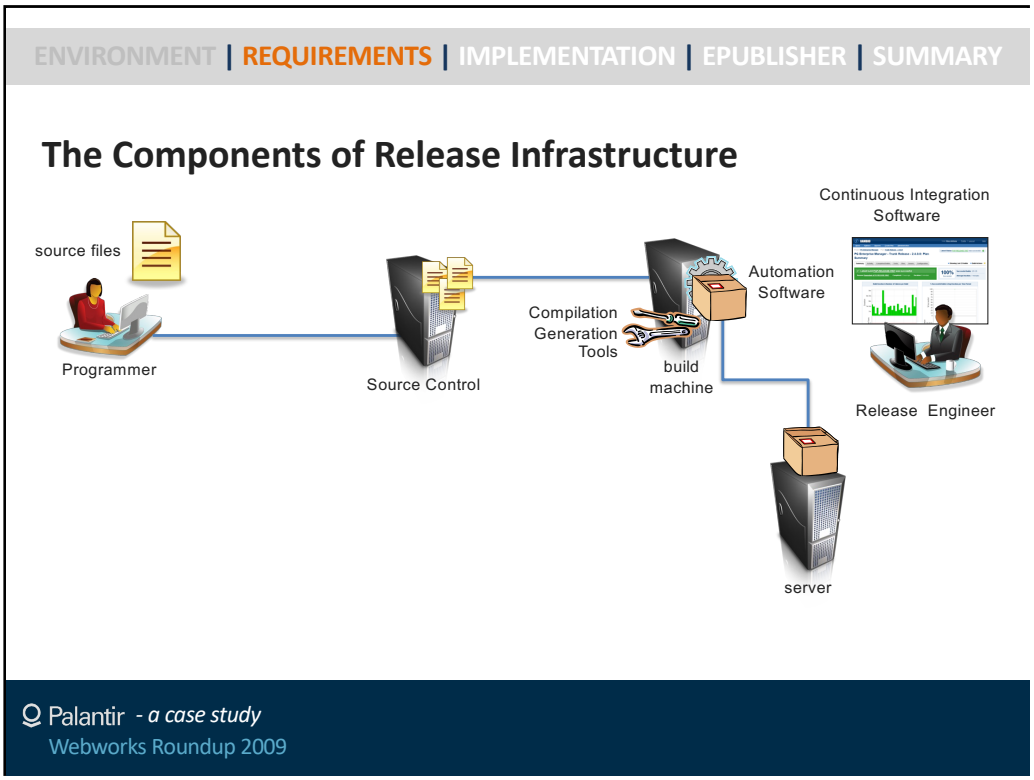
It succeeds most of the time so that other parts of the product can expect

Reportable

It is possible to collect and store historical data about your build.

This allows you to improve the system you are. You can achieve this by writing and maintaining a set of procedures

that you disseminate to writers but invariable this decreases characteristics of the process.



You need to understand the human and non-human components required to put together a good release engineering process:

Software release engineer

This is the software person responsible for putting together the software deliverable.

Documentation release engineer

This is the writer responsible for putting together the documentation deliverable.

Source Files

This the raw material that contains your code if you are software engineer. If you are documentation person, this is your Framemaker or other source.

Source change control system

Perforce, Clearcase, SVN

Branches or labels version of the source

Machine for running the build

Almost always a single machine and Windows

Build Automation software

Software that automates the steps for creating one or more documentation deliverables.

Ant, Mayven

Compilation or Generation software or tools

These are the pieces that the build automation calls on to convert the source to a deliverable.

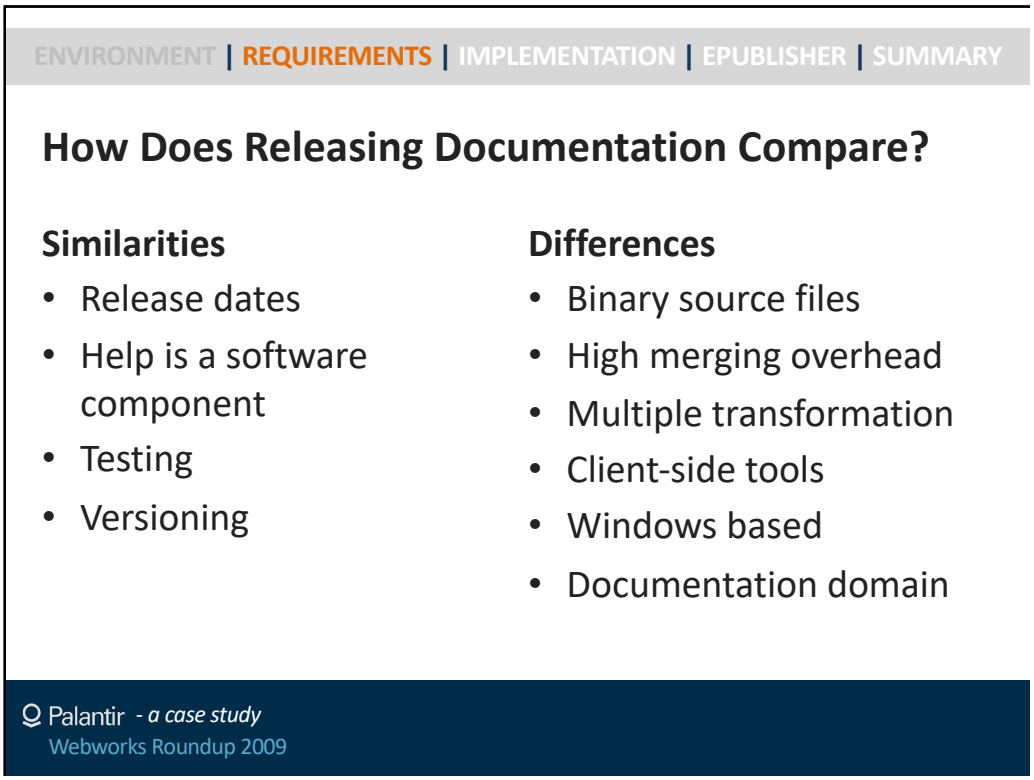
Dependency management software Ivy, Maven

javac,javadoc ePublisher Automap,

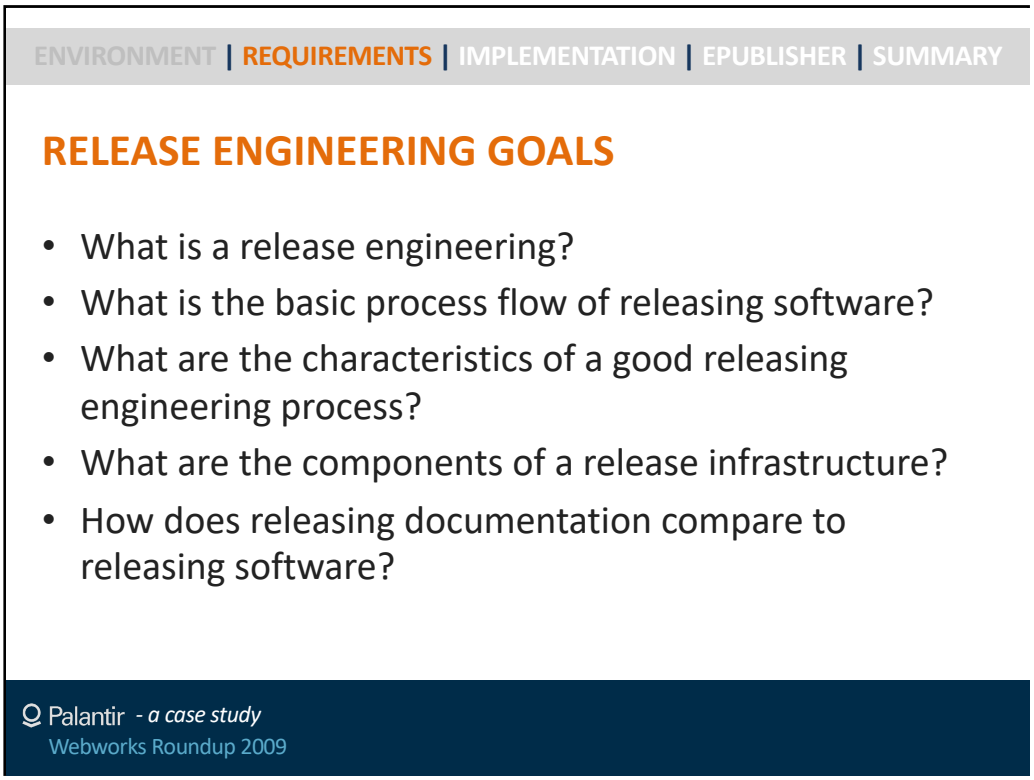
Continuous integration software

This is software that takes the output of your build and combines it with the output of others build to make a product.

Bamboo,



Software organizations understand building and managing software releases. In many ways a documentation build looks like a software build. In other ways it does not.



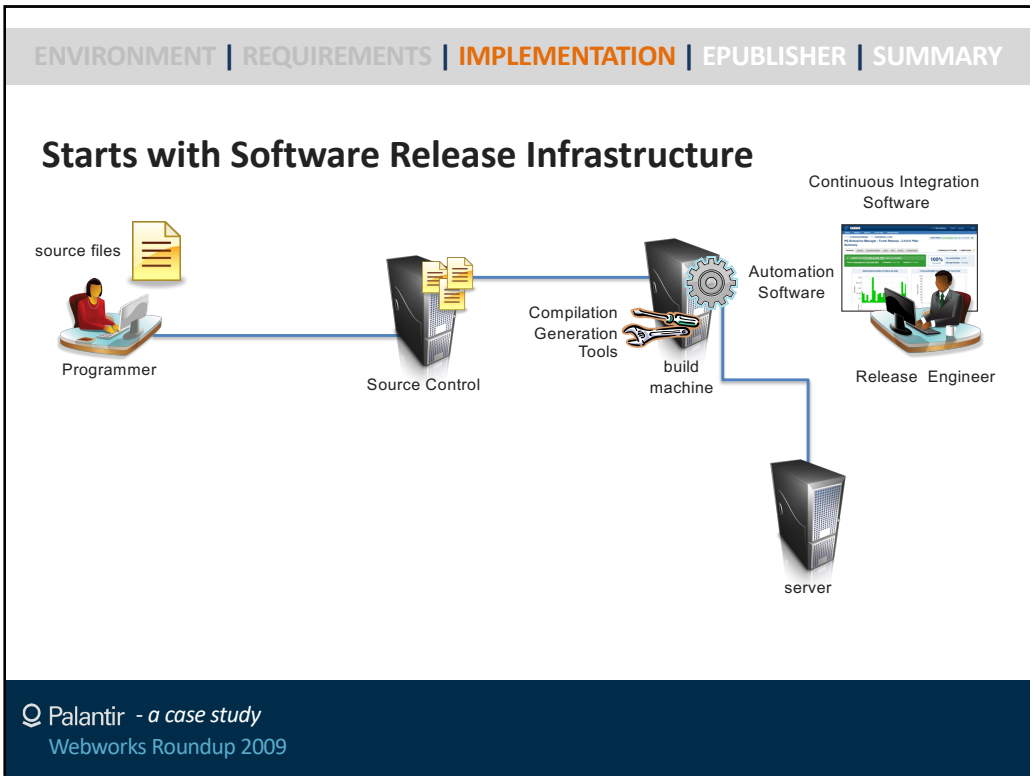
It is important to understand where the documentation build fits into the software development process. This understanding can help you:

- Make an argument for creating a build structure if you don't have one
- Find key people in your organization who can move your project forward
- Identify the goals of a good documentation build infrastructure

IMPLEMENTATION

This section takes a closer look at Palantir's documentation release engineering implementation.

- the component parts
- the software products used at each stage
- the inputs at each stage
- the out comes at each stage



You need to understand the human and non-human components required to put together a good release engineering process:

Software release engineer

This is the software person responsible for putting together the software deliverable.

Documentation release engineer

This is the writer responsible for putting together the documentation deliverable.

Source Files

This the raw material that contains your code if you are software engineer. If you are documentation person, this is your Framemaker or other source.

Source change control system

Perforce, Clearcase, SVN

Branches or labels version of the source

Machine for running the build

Almost always a single machine and Windows

Build Automation software

Software that automates the steps for creating one or more documentation deliverables.

Ant, Mayven

Compilation or Generation software or tools

These are the pieces that the build automation calls on to convert the source to a deliverable.

Dependency management software Ivy, Maven

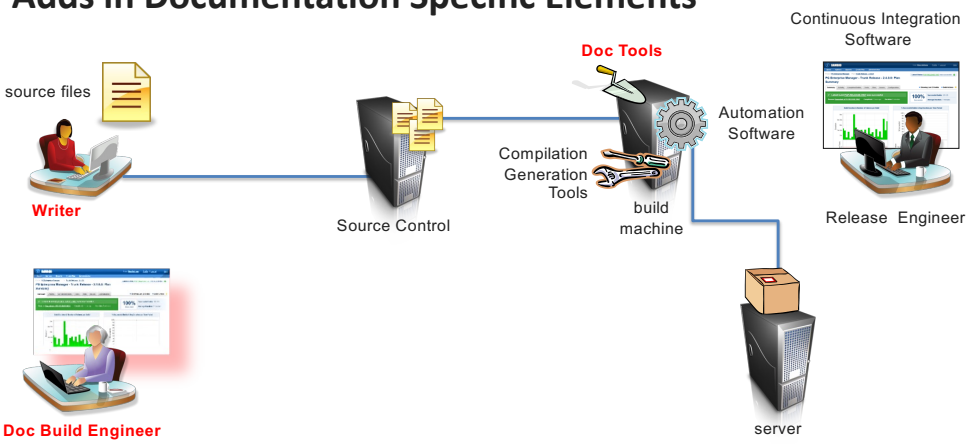
javac,javadoc ePublisher Automap,

Continuous integration software

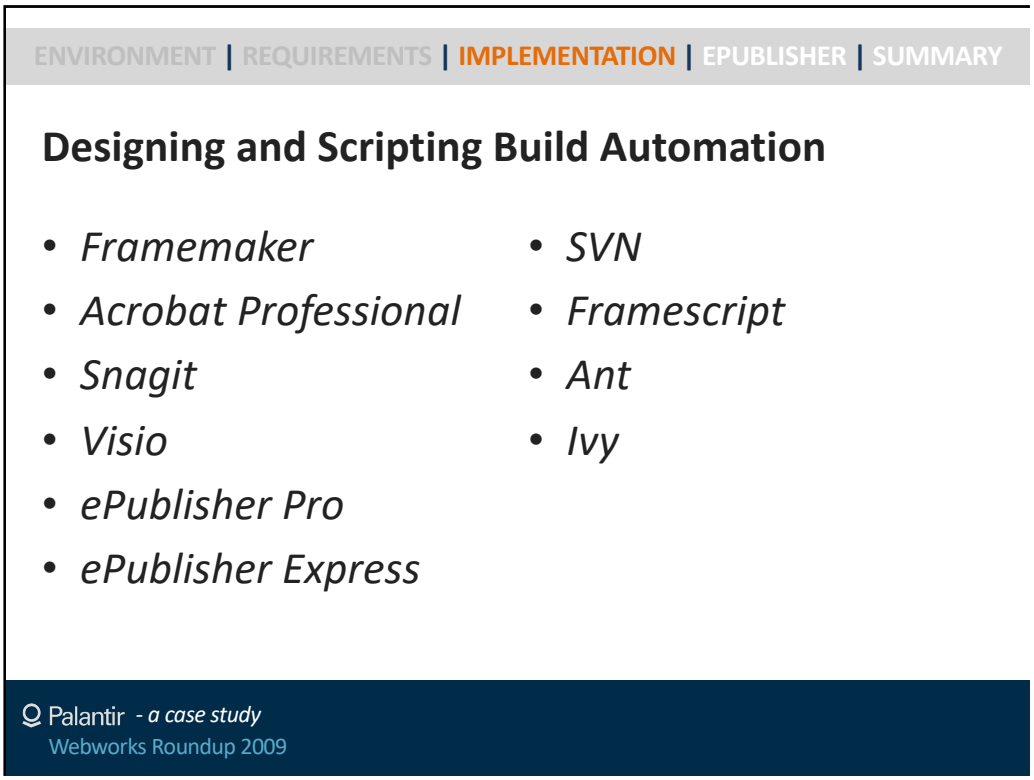
This is software that takes the output of your build and combines it with the output of others build to make a product.

Bamboo,

Adds in Documentation Specific Elements



Palantir's build structure uses the same tools as the software builds.
Additional tools are layered on that infrastructure.



1 person with a local machine

Responsibilities

•Designs content template in Framemaker

•Designs help and online formats using ePublisher Pro

•Generates stationery using ePublisher Pro

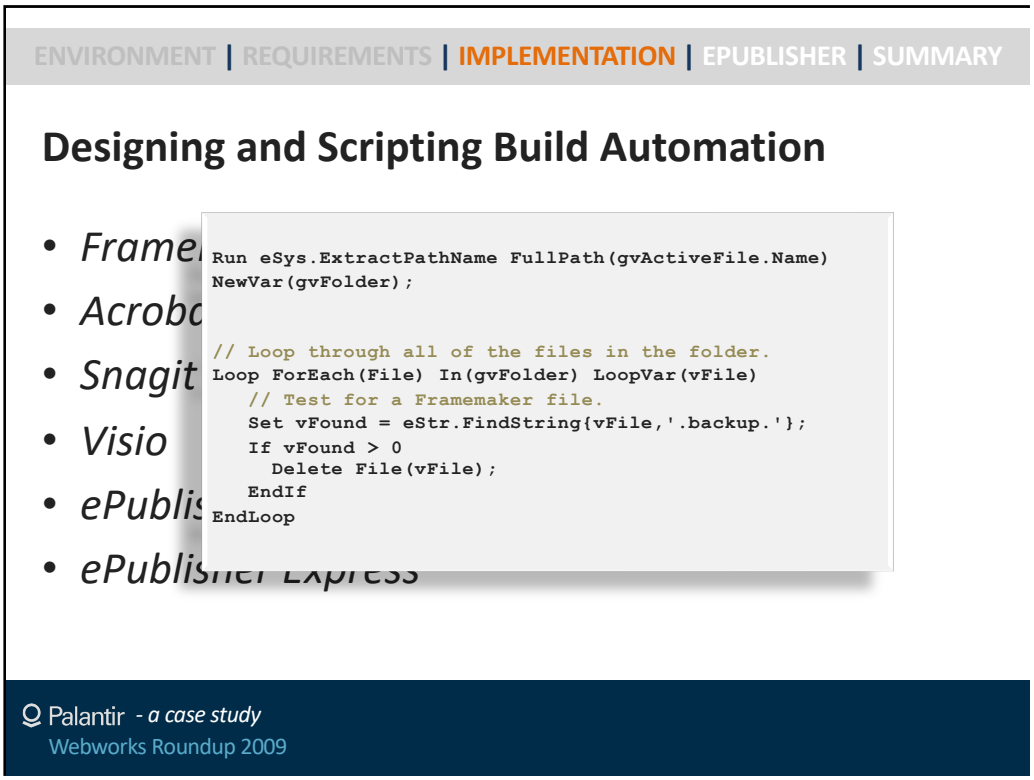
•Creates ePublisher projects with ePublisher Automap and an editor

•Writes Framescript that performs transformations on content.

•Writes Ant build scripts that automates the build.

•Maintains Ivy files that defines other software in the company dependent on our build.

•Stores the source for safe keeping in SVN.



1 person with a local machine

Responsibilities

•Designs content template in Framemaker

•Designs help and online formats using ePublisher Pro

•Generates stationery using ePublisher Pro

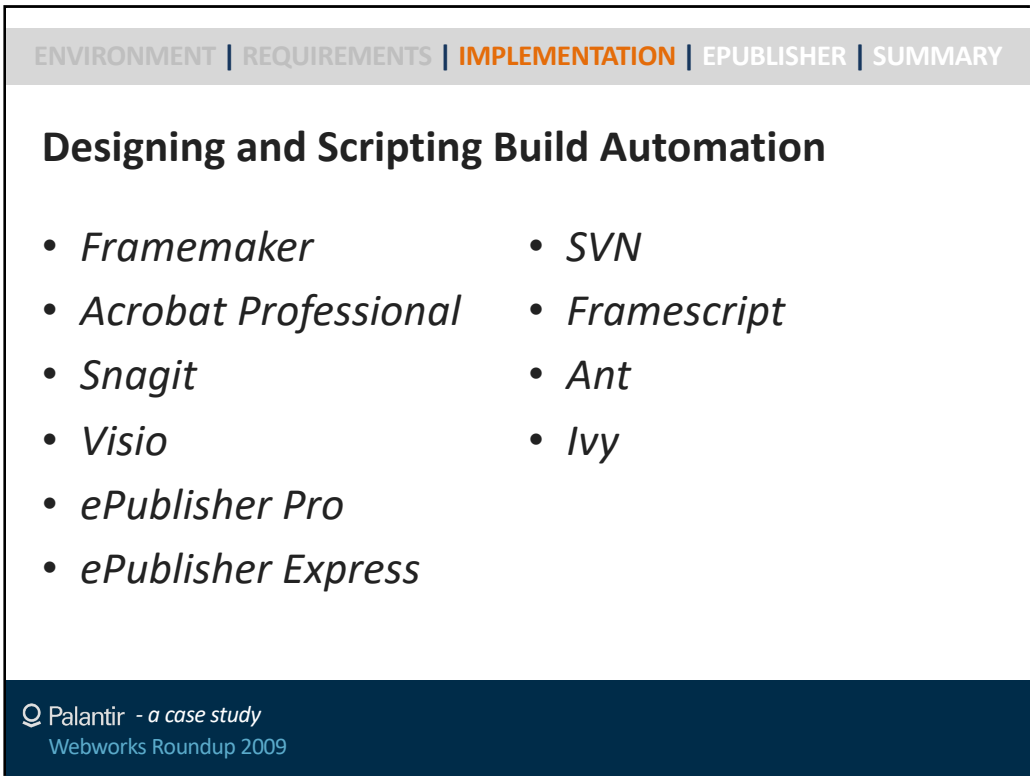
•Creates ePublisher projects with ePublisher Automap and an editor

•Writes Framescript that performs transformations on content.

•Writes Ant build scripts that automates the build.

•Maintains Ivy files that defines other software in the company dependent on our build.

•Stores the source for safe keeping in SVN.



1 person with a local machine

Responsibilities

•Designs content template in Framemaker

•Designs help and online formats using ePublisher Pro

•Generates stationery using ePublisher Pro

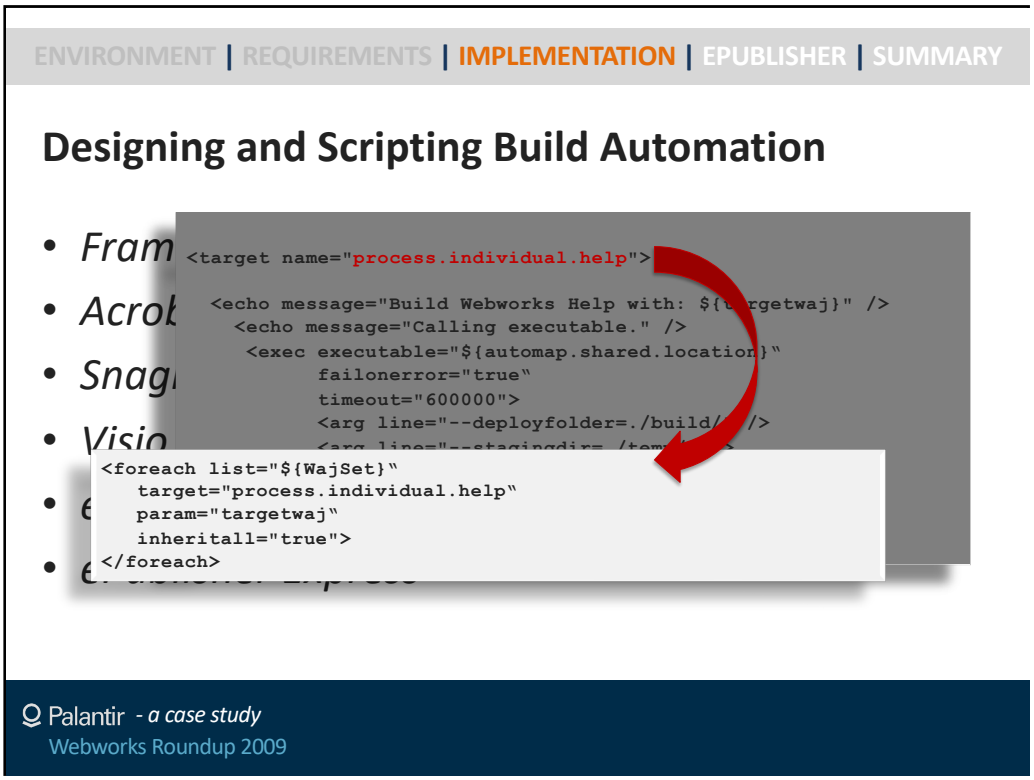
•Creates ePublisher projects with ePublisher Automap and an editor

•Writes Framescript that performs transformations on content.

•Writes Ant build scripts that automates the build.

•Maintains Ivy files that defines other software in the company dependent on our build.

•Stores the source for safe keeping in SVN.



1 person with a local machine

Responsibilities

•Designs content template in Framemaker

•Designs help and online formats using ePublisher Pro

•Generates stationery using ePublisher Pro

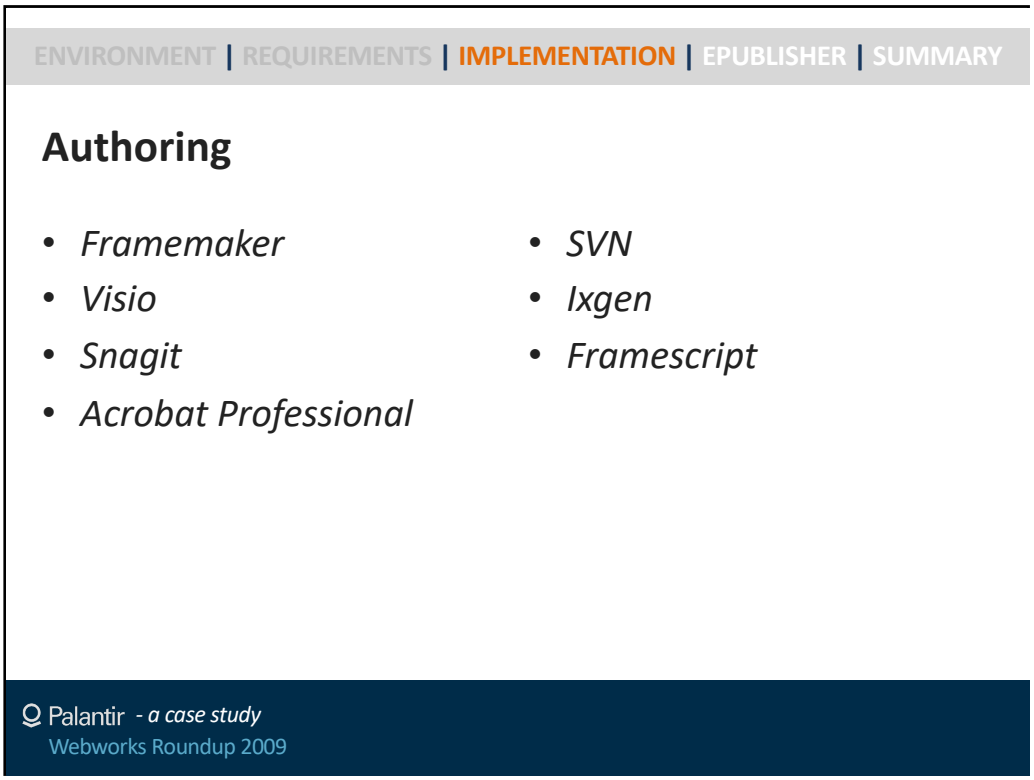
•Creates ePublisher projects with ePublisher Automap and an editor

•Writes Framescript that performs transformations on content.

•Writes Ant build scripts that automates the build.

•Maintains Ivy files that defines other software in the company dependent on our build.

•Stores the source for safe keeping in SVN.



4 writers with local machines

Responsibilities

•Produce content in Framemaker

•Screen captures in Snagit (PNG)

•Graphics in Visio (PNG)

Source Control Software

- Framemaker
- PNG
- Build scripts
- Framescripts
- Master project
- Stationery
- WAJ files

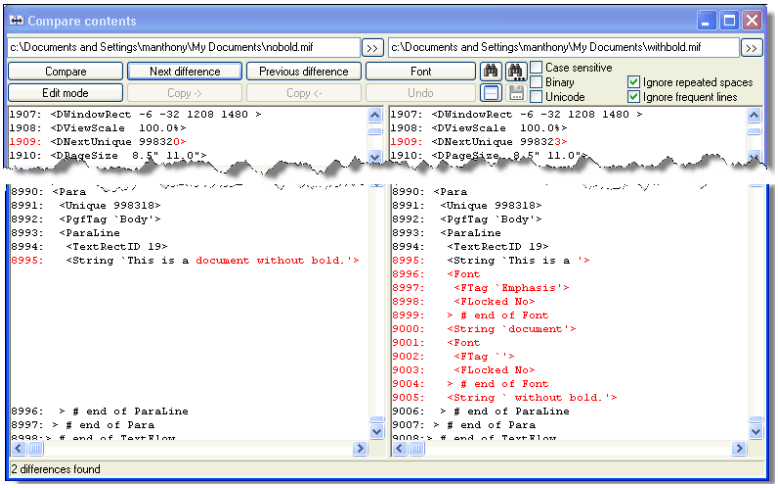
We use SVN this software stores the following sources for the build.

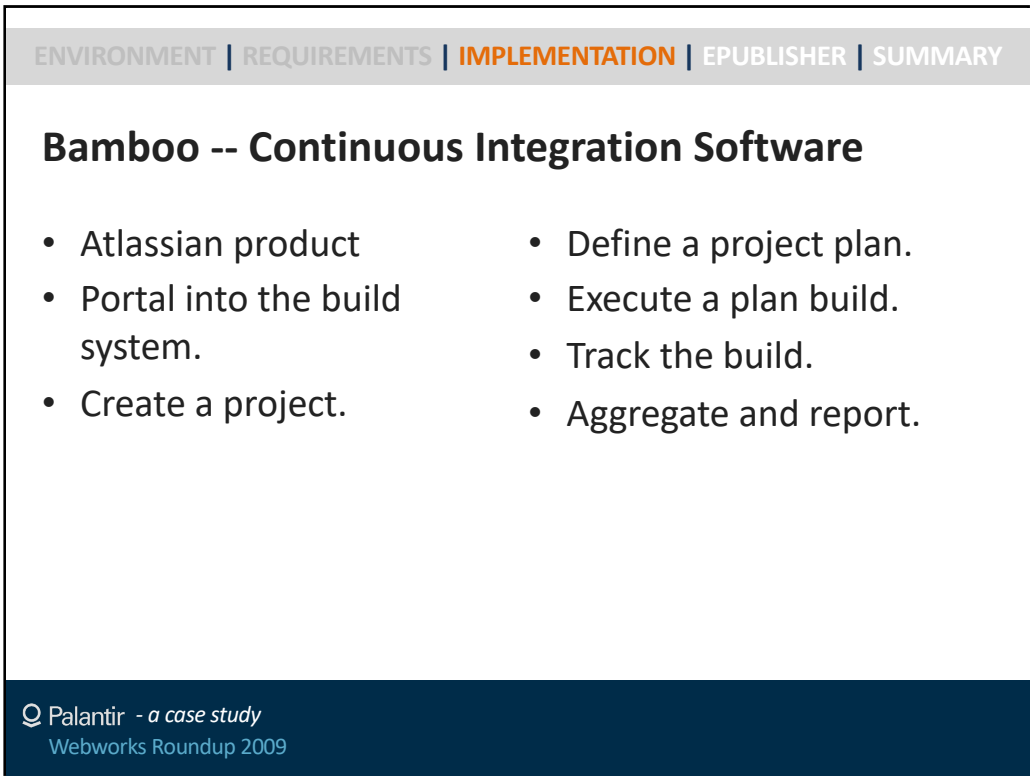
Why aren't we using MIF format so we can merge?

Filenames:

-- keep in mind where your files are stored. Unix file naming conventions. Directory Levels.

Merging Framemaker MIF?





Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

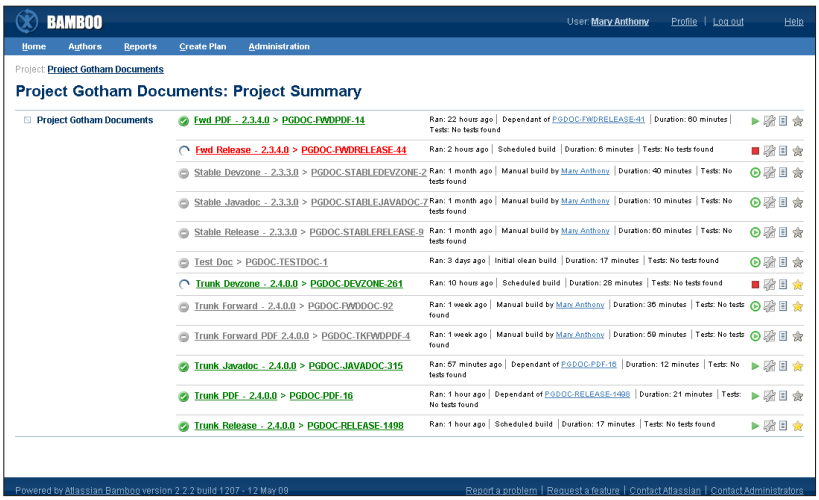
A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Ever plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Bamboo – Project Portal



Palantir - *a case study*
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

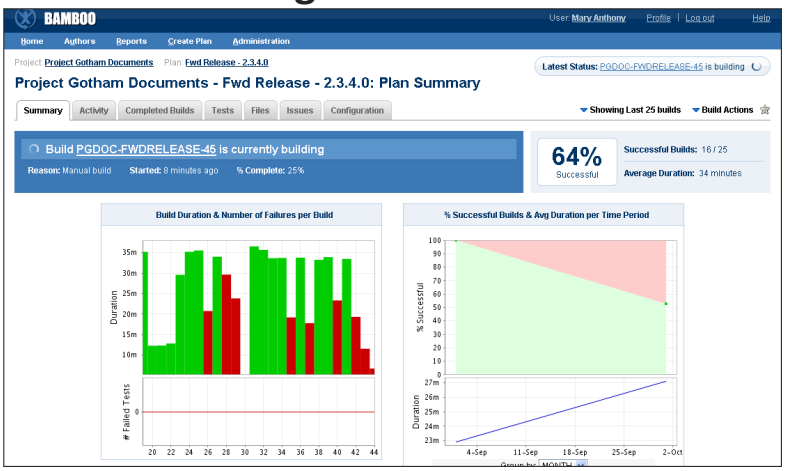
A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Ever plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Bamboo – Plan Page



Palantir - *a case study*
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

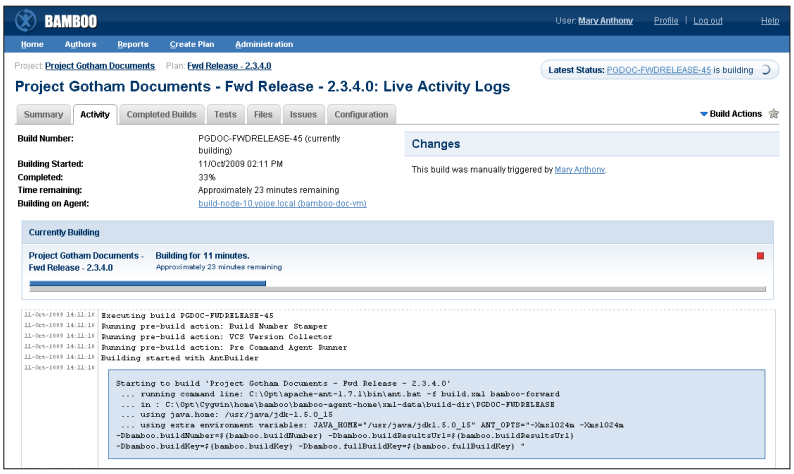
A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Ever plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Bamboo – Plan Page



Palantir - *a case study*
Webworks Roundup 2009

Bamboo 1 server, 16 remote agents, and 1 local agent. This machine:

Defines a plan which is a recipe for a build

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build ; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

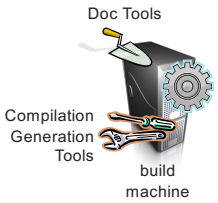
Ever plan belongs to a project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans

A build is one execution of a plan

Build Machine

- Windows machine
- Must meet ePublisher requirements
- Runs a Bamboo agent process



A Windows client machine. This machine:

Runs a Bamboo agent process: waiting to build

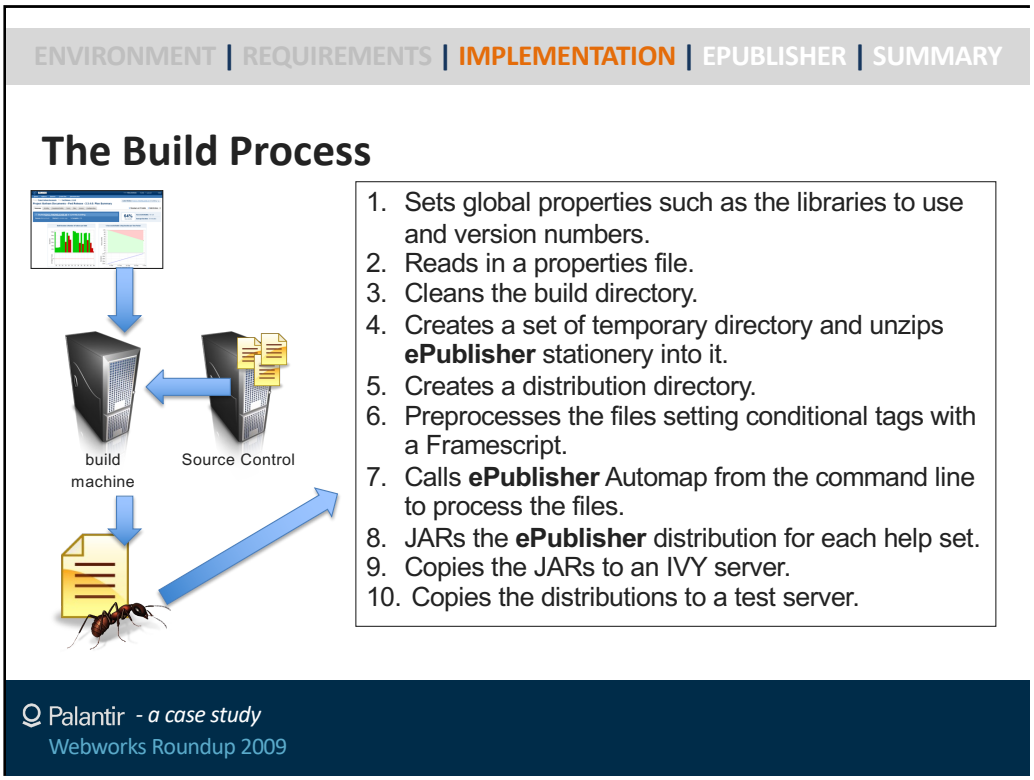
The agent process calls the Ant Build script

Ant tasks:

- Clean the directories from the previous build
- Setup the ePublisher stationery directories
- Preprocess the Framemaker files with Framescript
- Call ePublisher Automatp to generate the help and online deliverables.
- Call Adobe Acrobate to generate the PDF.
- Save the files to the Ivy repository
- Copy the build results to the test server

The Bamboo agent tracks the build and collects the log

The Bamboo agent collects and provides the artifacts from the build



A Windows client machine. This machine:

Runs a Bamboo agent process: waiting to build

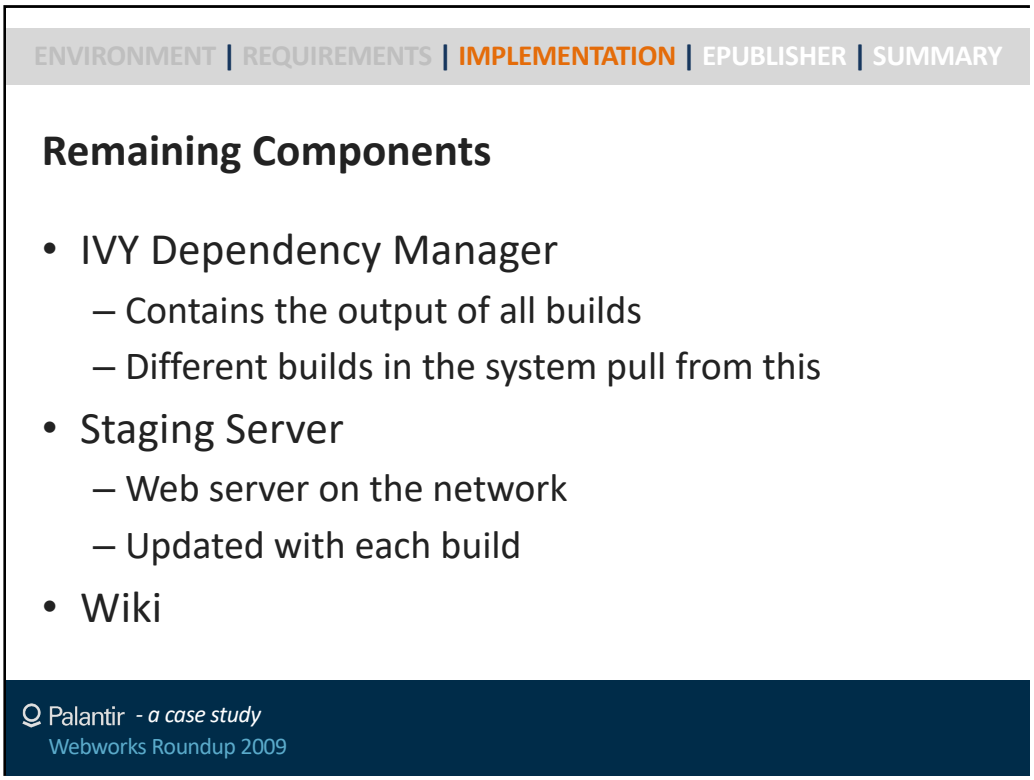
The agent process calls the Ant Build script

Ant tasks:

- Clean the directories from the previous build
- Setup the ePublisher stationery directories
- Preprocess the Framemaker files with Framescript
- Call ePublisher Automatp to generate the help and online deliverables.
- Call Adobe Acrobate to generate the PDF.
- Save the files to the Ivy repository
- Copy the build results to the test server

The Bamboo agent tracks the build and collects the log

The Bamboo agent collects and provides the artifacts from the build



IVY Dependency Manager

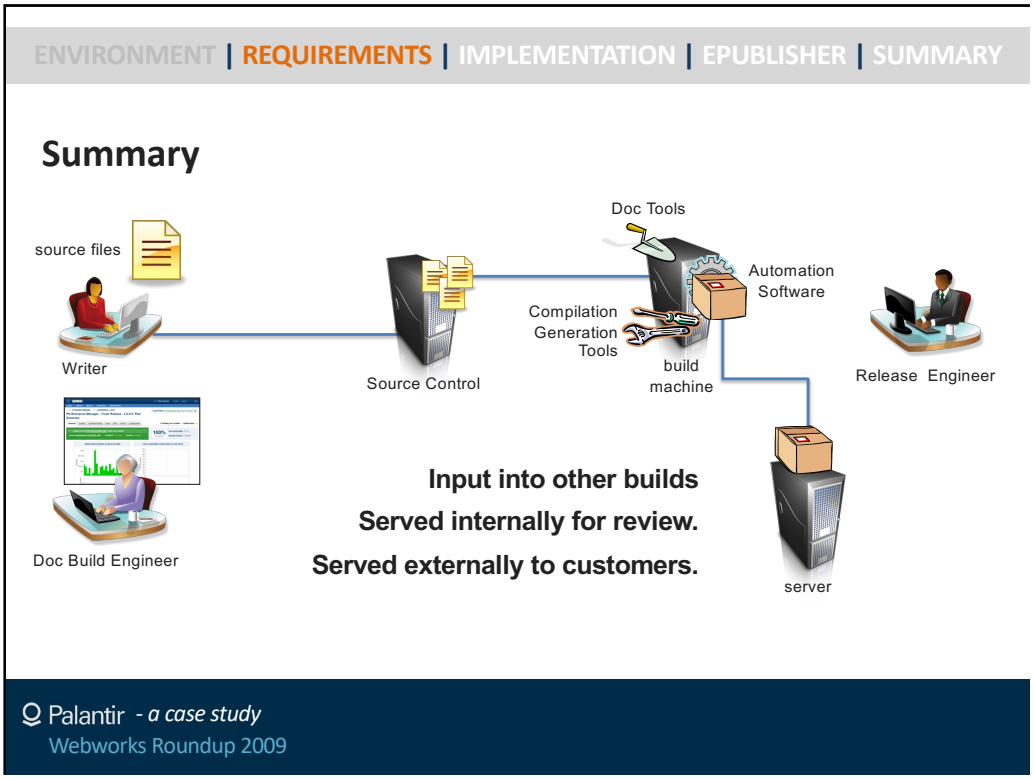
Receives the output of the build and serves it to other builds that require that output.

Staging Server

The Ant build copies the results of the build across the network to a web server.

Wiki

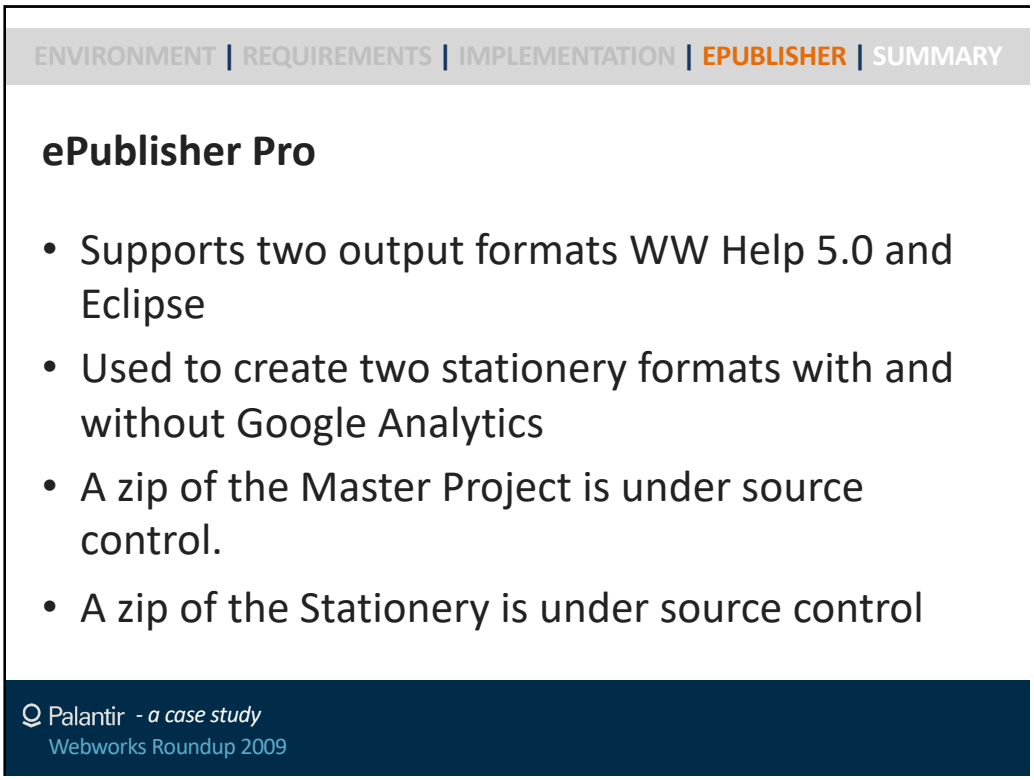
Provides links to the material on the web server.



Palantir's build structure uses the same tools as the software builds.
Additional tools are layered on that infrastructure.

EPUBLISHER

This section takes a closer look at each of the role of the ePublisher Software in Palantir's build infrastructure.



The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

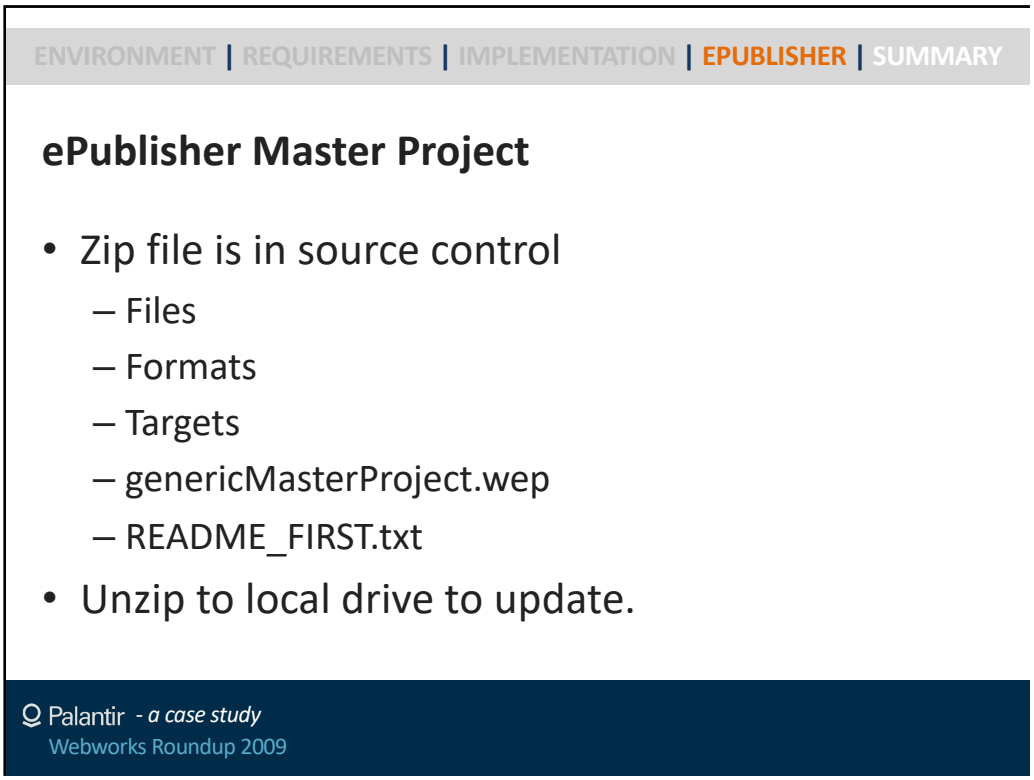
Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt



The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

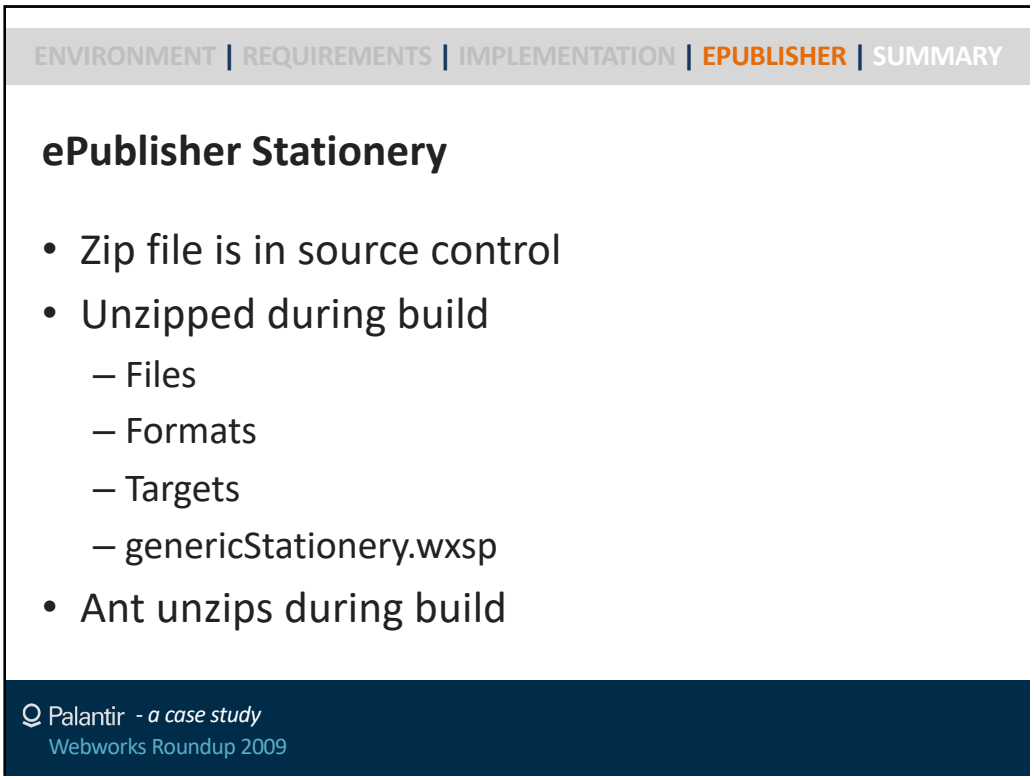
Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt



The designer/build works with the software to create a single Master Project. This Master Project:

Used to create two stationery formats with and without Google Analytics

Supports two output formats WW Help 5.0 and Eclipse

A zip of the Master Project is placed under source control.

Updated infrequently. Template changes and upgrades of the product.

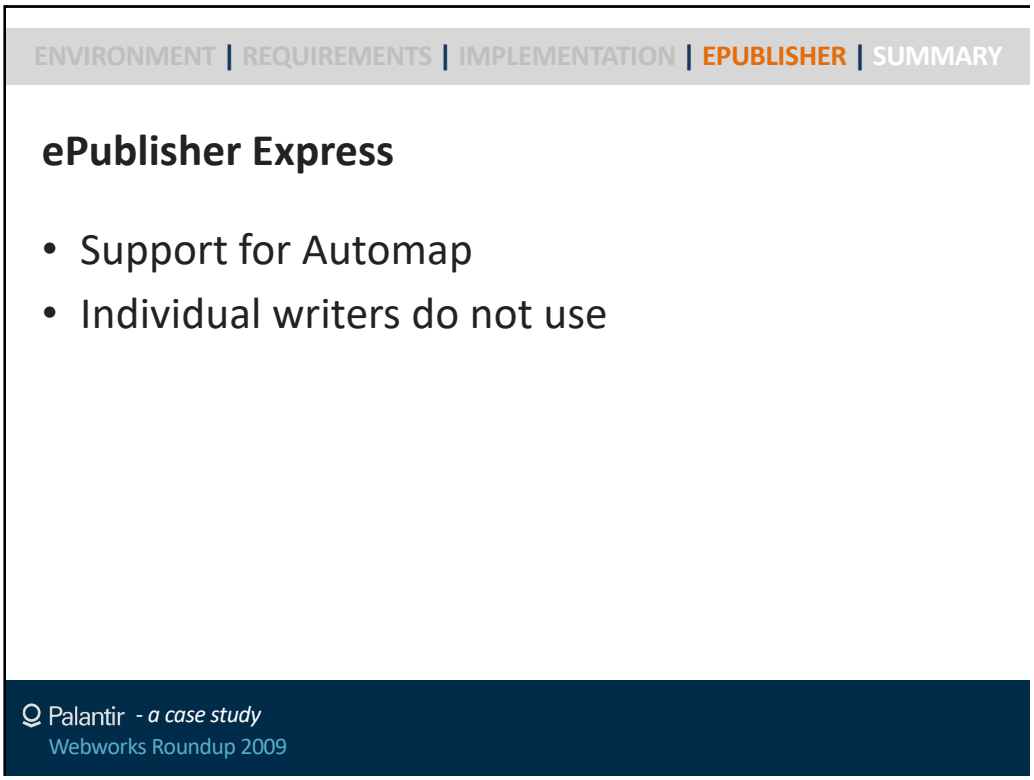
Files

Formats

Targets

genericMaster.wrp

README_FIRST.txt



Is used to support the Automapper. Our individual writers do not use this product.

Reasons for this decision:

•Reduces the number of tools writers need to learn. Too onerous given our release cycle

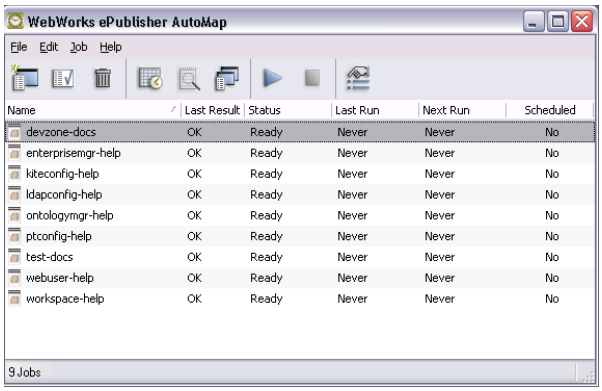
•Writers should focus on learning and writing about Palantir products.

•Writers are all located locally with easy access to the build machine.

•Writers review output from the frequent builds.

ePublisher Automap

- Registers projects
- Launched from the command line via Ant



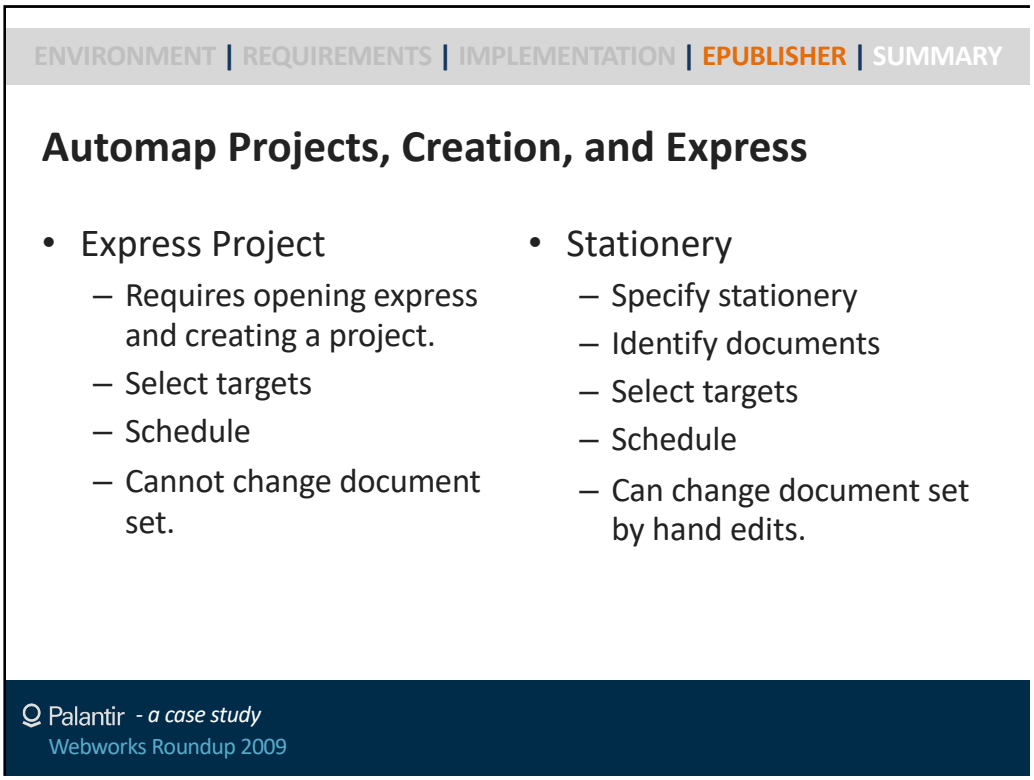
Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.



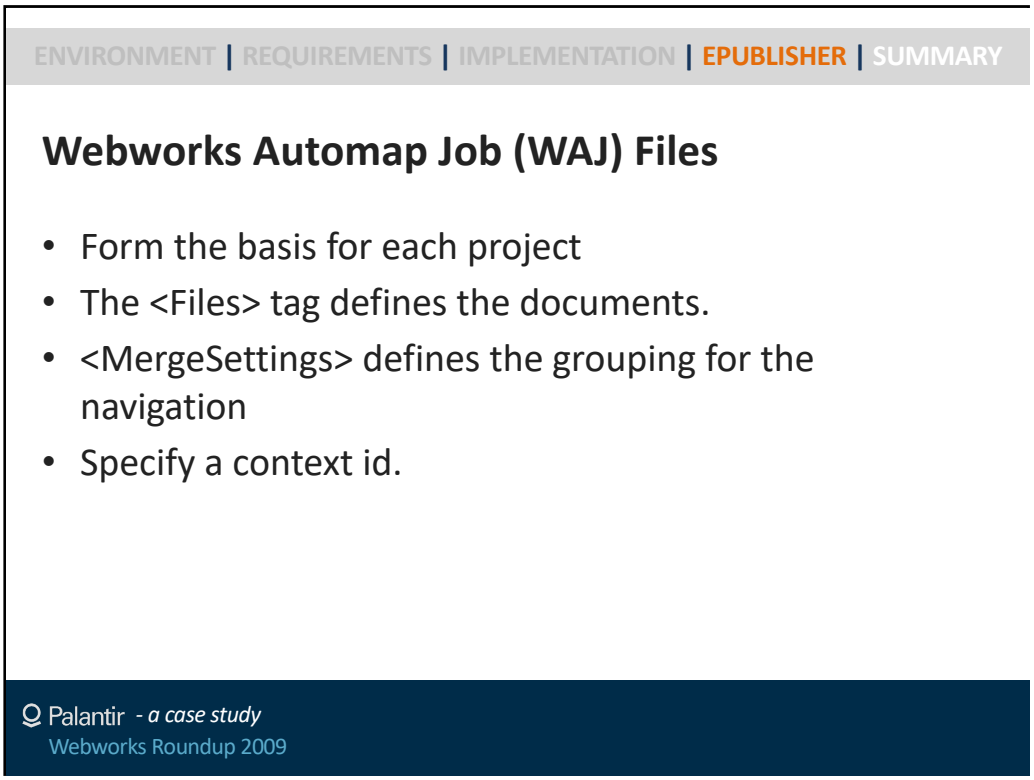
Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.



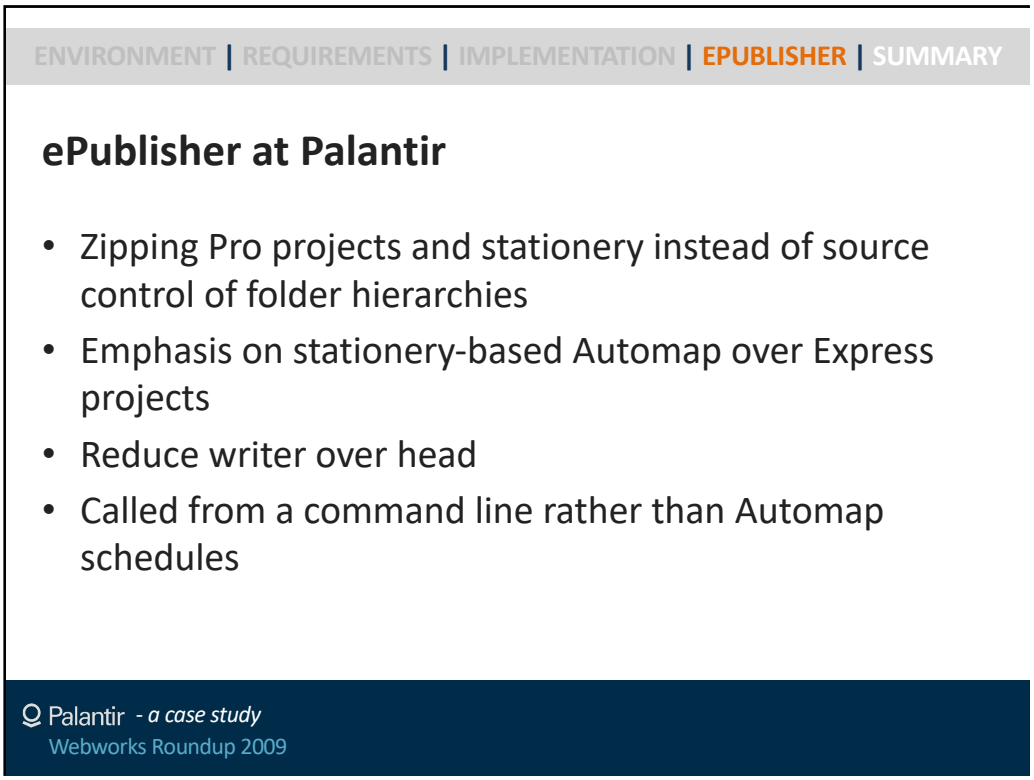
Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.



Is used to launch the documentation generation process.

WAJ files are registered through the Automapper but not constructed.

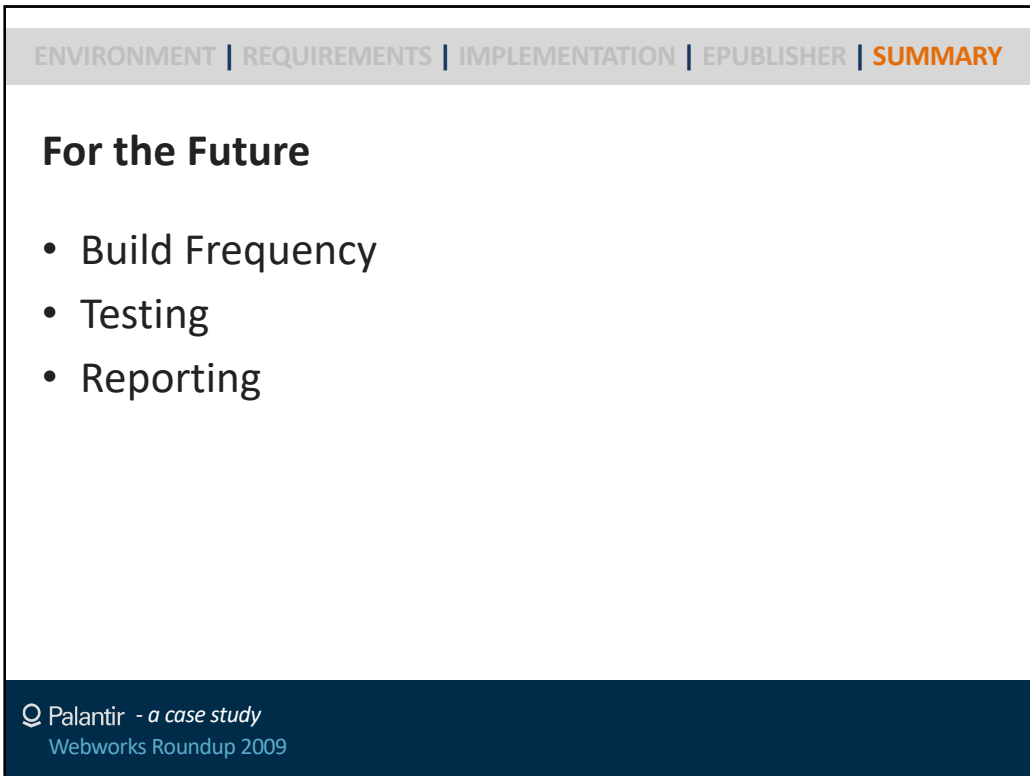
An editor is used to directly edit the WAJ file once it is registered with the Automapper.

Called from the Ant build process

We do not use the script feature of Automap. Keeping the build processes deconstructed allows us to easily debug problems.

SUMMARY

- This section discusses:
  - Areas we are seeking to improve
  - Pros and Cons



Build Frequency

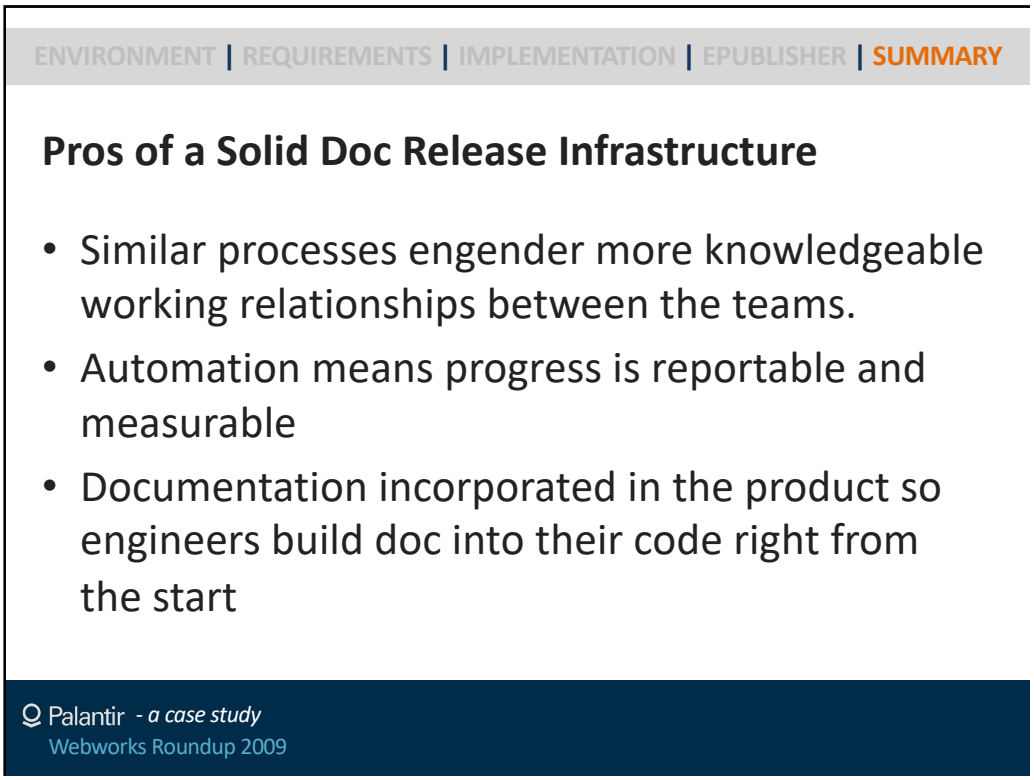
Want to provide a facility for individual writers to launch a personal build through our infrastructure. This will enable tighter build/review cycles near the end of a build iteration.

Reporting

Are not currently using the Reporting capability of ePublisher. This is a powerful tool we want to make use of. Time limitations not in terms of implementing but in terms of taking the results and managing them.

Testing

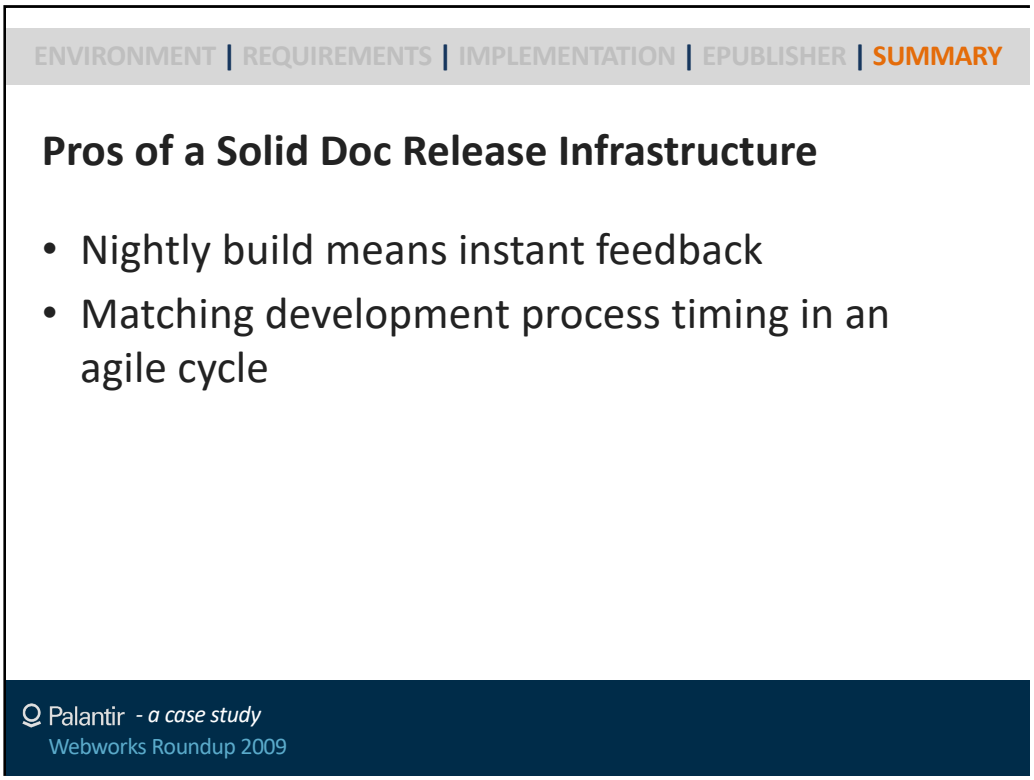
Writing scripts that test builds before they are submitted.



For example, when a new release is anticipated, because we are “in the loop” with the release engineering team, we have a faster track as writers to hearing about it.

By knowing how the release engineers work, we can foster improvements in their processes and speed the delivery of our documentation process.

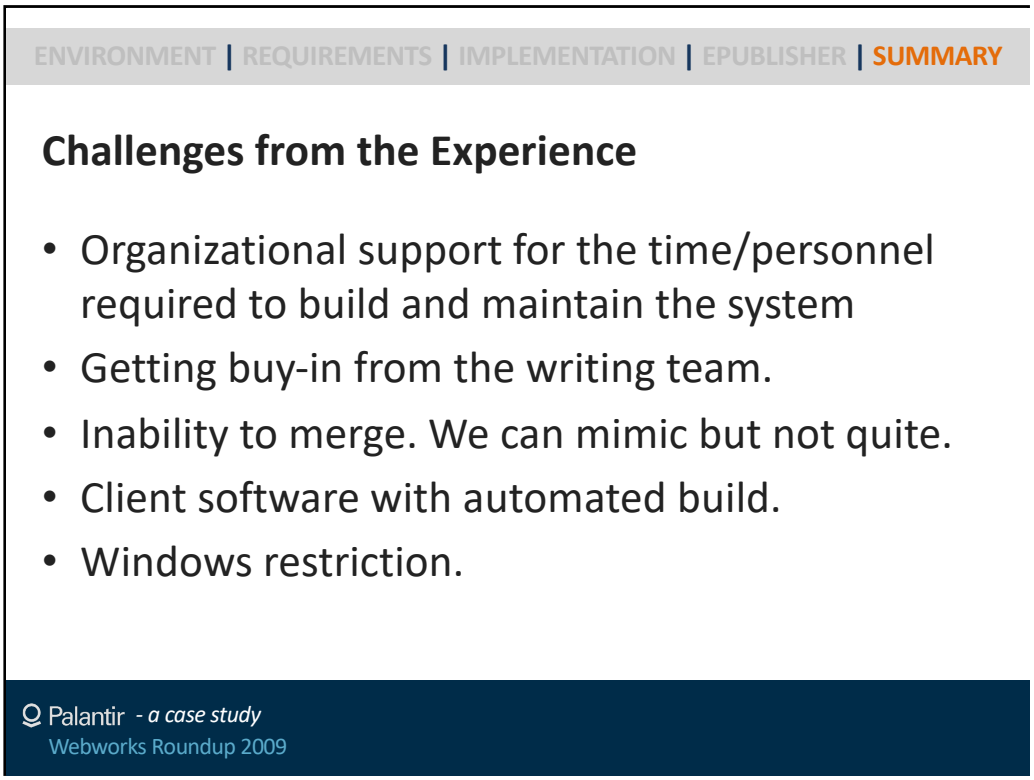
Engineers learn what goes into producing the documentation.



For example, when a new release is anticipated, because we are “in the loop” with the release engineering team, we have a faster track as writers to hearing about it.

By knowing how the release engineers work, we can foster improvements in their processes and speed the delivery of our documentation process.

Engineers learn what goes into producing the documentation.



Documentation builds take knowledge about documentation products, processes, and scheduling. It takes a person who is both a writer and a bit of an engineer.

Writing team can be resistant to the processes automation. Requires new tools and new ways of doing things.

Positive Process Impacts

- Nightly build means instant feedback
- Matching development process timing in an agile cycle

Acknowledgements

- Richard Neale, Palantir Release Engineer

  "Thank you Obi-wan"

- Vicky Bialas & Liz Keene, Arcsight Doc Build Team

  "Thank you both for pithy comments in reviewing my presentation. And for sharing your own experience and knowledge."

QUESTIONS