Software Release Planning (SRP)

What is Software Release Planning

- The purpose of release planning is to develop a plan for delivering sequential increments of software product product.
- It’s a collaborative effort, involving:
  - Product Owners
  - Development team
  - End users
  - Any other stakeholder

Why Release Planning

- Software Release planning (SRP) addresses decisions related to selecting and assigning features to create a sequence of consecutive product releases that satisfies important technical, resource, budget, and risk constraints.

- Incremental development provides customers with parts of a system early, so they receive both a sense of value and an opportunity to provide feedback early in the process. Each system release is thus a collection of features that the customer values.

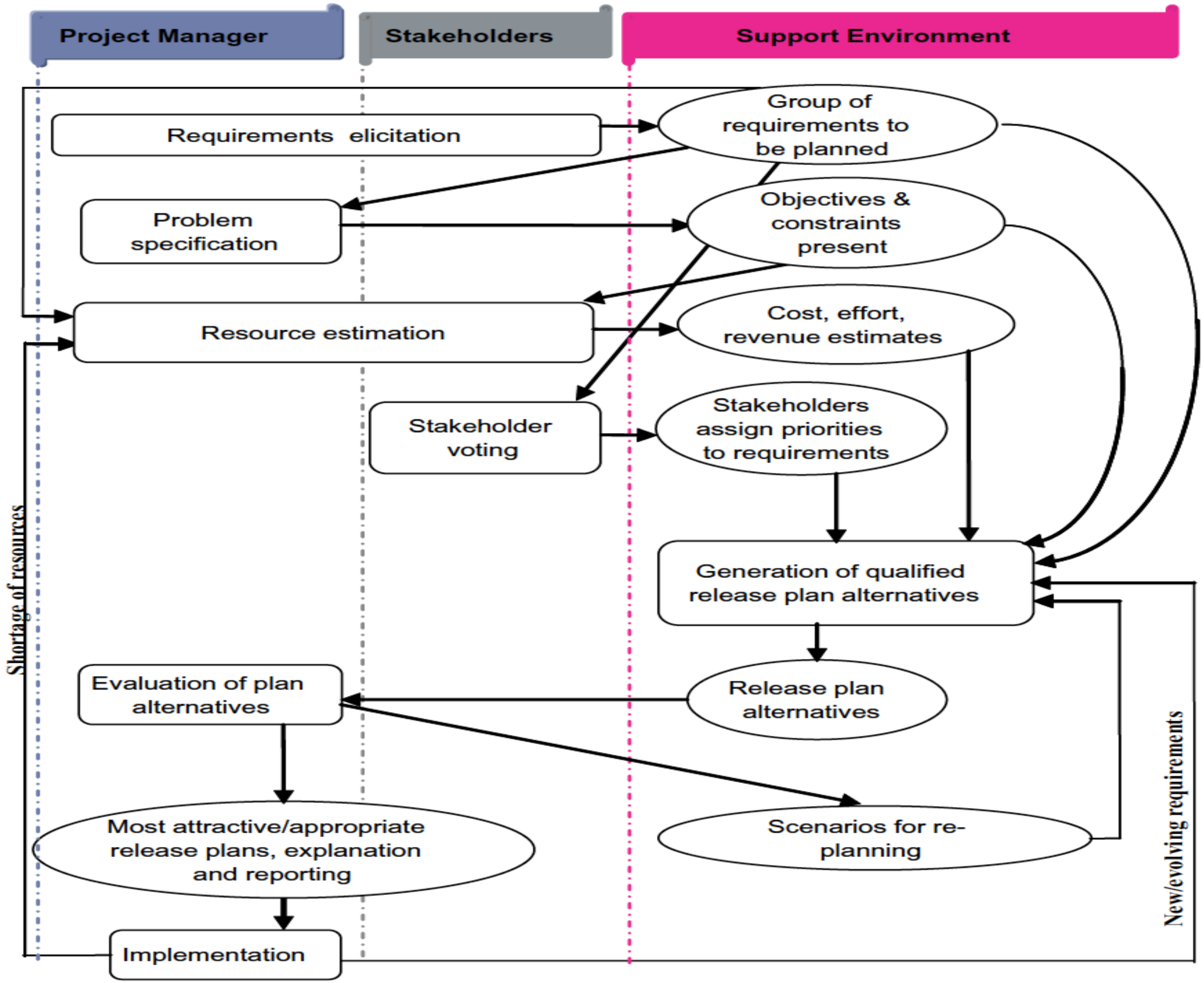
Good SRP Should:

- provide maximum business value by offering the best possible blend of features in the right sequence of releases,
- Satisfy the most important stakeholders involved,
- Be feasible with available resources, and reflect existing dependencies between features.

An Integration of Art and Science

- The art of SRP approach relies on human intuition, communication, and capabilities to negotiate between conflicting objectives and constraints.
- The science of RP approach formalizes the problem and applies computational algorithms to generate best solutions.
- At the industrial level we need to create a synergy between the two, integrating human and computational intelligence to define optimal RP feature assignments.

A possible Process Model for SRP



From: J. Momoh, G. Ruhe, Release Planning Process Improvement – An Industrial Case Study

SRP Challenges

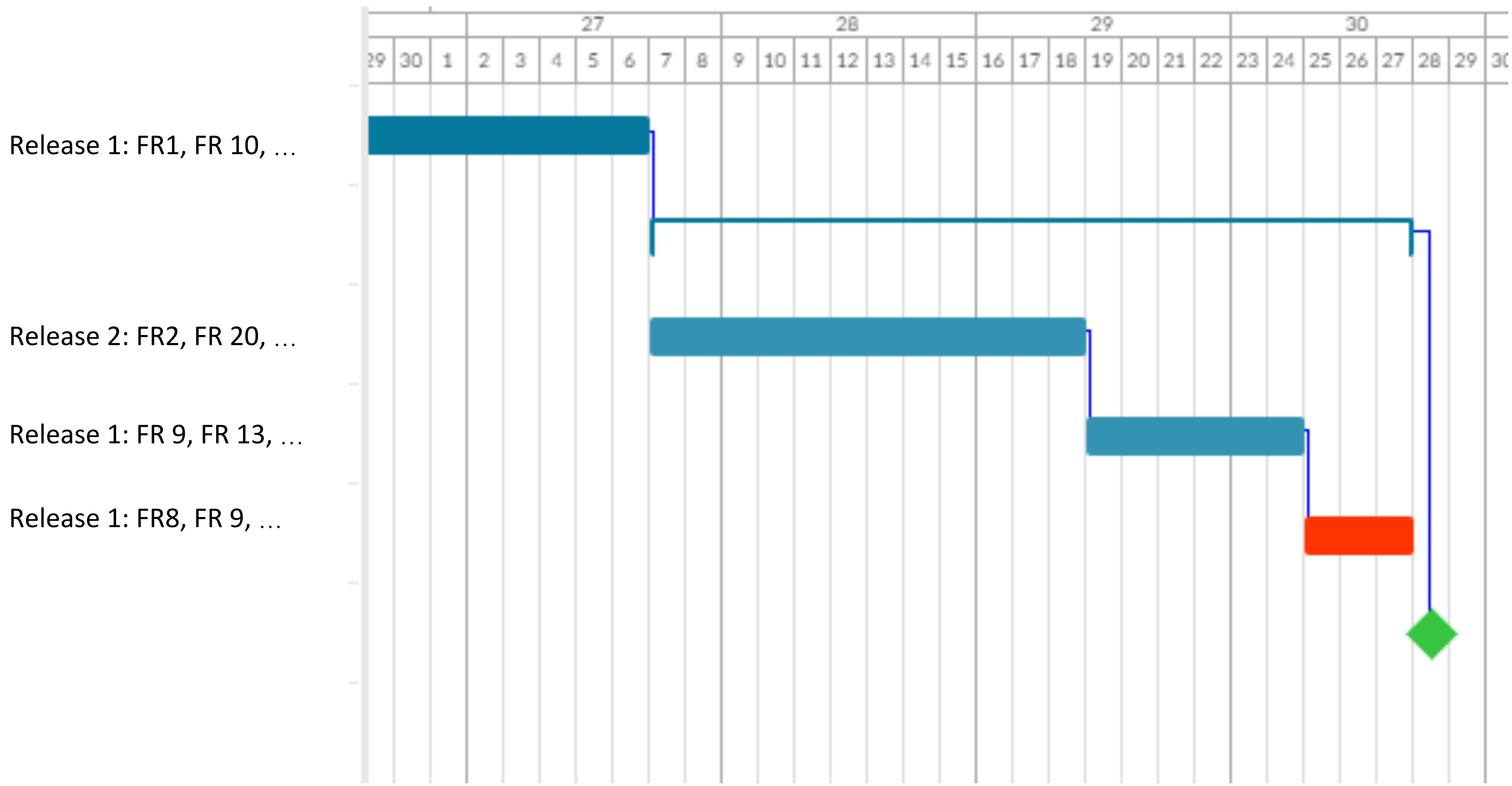
- You need a proper understanding of the planning objectives and constraints as well as of the important stakeholders and their feature preferences.
- The number of conflicting features in complex projects can be the biggest challenge. In those cases you need a release planning tool with a optimization algorithm.

Release Planning Requirements

- High level input about business objectives
  - Ranked business goals
- Input about development
  - Skill level
  - Previous experience
- Technical input
  - Technology risks
  - Architectural dependencies
- Input from business owner:
  - Possible business risks
  - Time to market concerns
- List of prioritized requirements
  - Business level
  - Product level

Expected Outcome

- A high-level development process Gantt Chart



Requirement Prioritization

Requirements Prioritization

- Sometimes there is conflict between client’s requirements:
  - You may need establish a compromise between conflicting requirements.
- There are different techniques to conduct a comparison between requirements:
  - Simple Voting Approach
  - Cost Value Approach
  - Systematic Decision Grid
  - Pairwise Comparison Chart
  - Eigen Vector (a more advanced method of assessment)

Simple Stakeholders Voting Approach

- High – the requirement is a must-have and critical to the stakeholder business;
- Medium – the business can still be carried on without the requirement, with some workaround;
- Low – the requirement would be nice to have.

Other Techniques

Number of Comparisons

- Number of comparisons depends on the number of requirement items to compare:
- # of requirement / # of Comparison

# of requirement	# of Comparison
2	1
3	3
4	6
5	10
6	15

- Or, in general we can say:

  # of comparisons = $n(n-1)/2$

Adopted from: Dr. Kardi Teknomo Slides

Cost Value Approach

- A good and relatively easy to use method for prioritizing software product requirements is the cost-value approach.
- The basic idea is to determine for each individual requirement what is the cost of implementing the and how much value the requirement has.

Systematic Decision Grids

- A Systematic Decision Grid is a simple table that lists the features/behaviors relevant to a project/problem on each column, and each goal listed on the subsequent rows
  - cells of the table are scores or ranks

Features	A	B	C	*D*	E	Total
Goal 1	2	3	1	1	1	8
Goal 2	1	2	2	2	2	9
Goal 3	3	1	3	3	3	13

Pairwise Comparison Chart (PCC)

- Another common technique is to use a pairwise-comparison-chart (PCC).
- PCC uses a table as follows:
  - Table rows and columns are objectives
  - Each cell of the table indicates a binary value (0..1) for each pairwise comparison of objectives.

A Binary Approach

- Let's assume a system requires four competing features: performance, portability, reliability, security.

Goals	performance	portability	Reliability	security	Total score
performance	…	1	0	1	2
portability	1	…	1	1	3
Reliability	0	1	…	1	2
security	1	1	1	…	3

- In this example portability and security have the higher rank, and performance and reliability have the lower-rank, but it doesn't necessarily mean that those with lower ranks must be discarded.

An Aggregated Pairwise Approach

- In many cases there are more than one analyst and developers are involved in the prioritization process, and each member may have a different judgment.
- An Aggregated Pairwise Comparison Chart (APCC) can resolve this issue.
- Example:
  - Assume three engineers participate. Each engineer gives a binary vote

Goals	Performance	portability	Reliability	Security	Total score
Performance	…	0 + 0 +0	0 +1+0	1 + 1 +0	3
portability	1 + 0 + 1	…	1 + 1 + 1	1 + 1 + 1	8
Reliability	1 + 0 + 1	0 + 1 +1	…	1 + 1 + 0	6
Security	0 + 1 + 1	0 + 0 + 0	0 + 0 + 0	…	2

A None Binary Approach

- Another approach is a None Binary
- Example:
  - Engineers discuss all the details and will come of with scores, based on some pre-defined rang of values (say 0 to 5)

Goals	Performance	portability	Reliability	Security	Total score
Performance	…	3	0	4	7
portability	5	…	5	3	13
Reliability	1	3	…	4	8
Security	0	0	4	…	4

Using Eigen Vector

Using Eigen Vector

- It was originally developed by Prof. Thomas L. Saaty.
- Uses paired comparisons.
- The ratio scales can be derived from some mathematical method:
  - Eigen vectors and the consistency index derived from the principal Eigen value.
- It doesn’t prescribe any decision, but helps decision makers to find one the best.

A Simple Example

Apply a Pairwise Ratio

Given: Three different feature with three different values.

Feature A V1 = 100
Feature B V2 = 50
Feature C V3 = 25

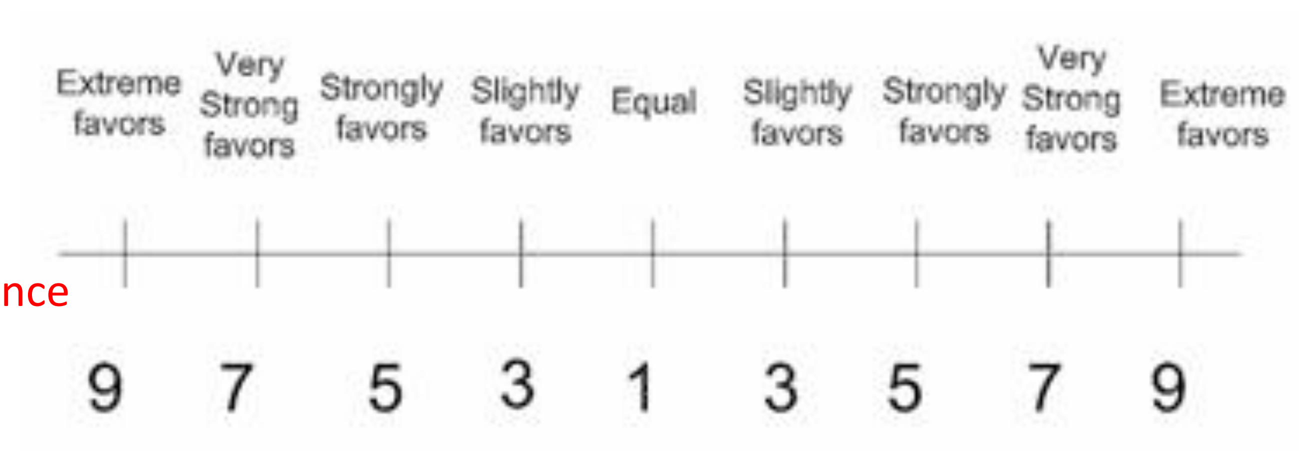
Value Comparison	**A**	**B**	**C**
A	V_1/V_1	V_1/V_2	V_1/V_3
B	V_2 / V_1	V_2 / V_2	V_2 / V_3
C	V_3 / V_1	V_3 / V_2	V_3 / V_3

Adopted from: Dr. Kardi Teknomo Slides

How to Set Values

- Which feature is more important to you

Better Performance



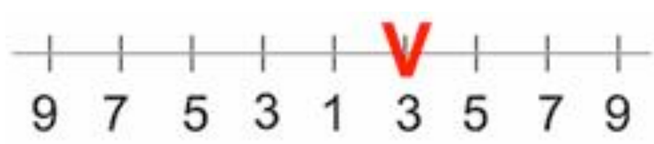
More Security

Figure from: Kardi Teknomo Slides

Example:

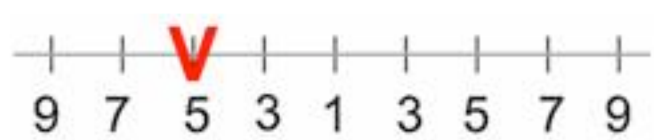
– assume we have only three kinds of requirements to be compared: Safety, Security, Reliability

Reliability



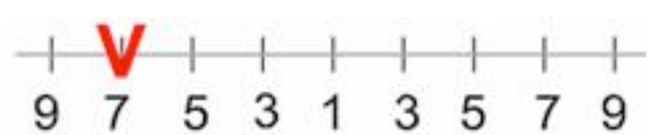
Safety

Reliability



Security

Safety



Security

Adopted from: Dr. Kardi Teknomo Slides

Build a matrix

- Transfer the result into a matrix

$$A = \begin{array}{lc} & \begin{array}{ccc} \underline{\textbf{Rel.}} & \underline{\textbf{Safe.}} & \underline{\textbf{Sec.}} \end{array} \\ \begin{array}{l} \text{Reliability} \\ \text{Safety} \\ \text{Security} \end{array} & \begin{Bmatrix} 1 & 1/3 & 5 \\ & 1 & 7 \\ & & 1 \end{Bmatrix} \end{array}$$

Adopted from: Dr. Kardi Teknomo Slides

Fill the lower triangular

- Use the reciprocal values of the upper diagonal.
- If a_{ij} is the element of row column of the matrix, then the lower diagonal will be:

$$a_{ji} = 1/a_{ij}$$

$$A = \begin{array}{c c} & \begin{array}{ccc} \textbf{Rel} & \textbf{Safe} & \textbf{Sec} \end{array} \\ \begin{array}{c} \text{Rel} \\ \text{Safe} \\ \text{Sec} \end{array} & \left[\begin{array}{ccc} 1 & 1/3 & 5 \\ 3 & 1 & 7 \\ 1/5 & 1/7 & 1 \end{array} \right] \end{array}$$

Adopted from: Dr. Kardi Teknomo Slides

Sum each column

- We sum each column of the reciprocal matrix to get

$$
A = \begin{array}{cc} & \begin{array}{ccc} \textbf{Rel} & \textbf{Safe} & \textbf{Sec} \end{array} \\ \begin{array}{c} \text{Rel} \\ \text{Safe} \\ \text{Sec} \end{array} & \begin{bmatrix} 1 & 1/3 & 5 \\ 3 & 1 & 7 \\ 1/5 & 1/7 & 1 \end{bmatrix} \end{array}
$$

Sum = 21/5 31/21 13

Adopted from: Dr. Kardi Teknomo Slides

Normalize the matrix

- Divide each element of the matrix with the sum of its column.

$$A = \begin{array}{c} \\ \text{Rel} \\ \text{Safe} \\ \text{Sec} \end{array} \begin{array}{c} \underline{\textbf{Rel} \quad \textbf{Safe} \quad \textbf{Sec}} \\ \left[\begin{array}{ccc} 5/21 & 7/31 & 5/13 \\ 15/21 & 21/31 & 7/13 \\ 1/21 & 3/31 & 1/13 \end{array}\right] \end{array}$$

$$\textbf{Sum} = \quad \textbf{1} \quad \textbf{1} \quad \textbf{1}$$

Adopted from: Dr. Kardi Teknomo Slides

Create an Eigen Vector

- The normalized principal Eigen vector can be obtained by averaging across the rows

$$\frac{1}{3}\left\{\begin{matrix} 5/21 & 7/31 & 5/13 \\ 15/21 & 21/31 & 7/13 \\ 1/21 & 3/31 & 1/13 \end{matrix}\right\} = \left\{\begin{matrix} 0.2828 \\ 0.6434 \\ 0.0738 \end{matrix}\right\}$$

Adopted from: Dr. Kardi Teknomo Slides

What is Priority Vector

- The normalized principal Eigen vector is also called priority vector .
- Sum of all elements in priority vector is 1. It shows relative weights.
- In this example:
  - Reliability is 28.28%
  - Safety is 64.34%
  - Security is 7.38%.
- Our most preferable requirement is Safety, followed by Reliability and Security.

From: Dr. Kardi Teknomo Slides

Another Example

- Problem Statement:
- *In a smart phone OS development , there were over 35 requirements that only a subset of those were supposed to be implemented in the next release. For the purpose simplicity, only a few of those requirements are listed here.*
  - Download response time vs start up response = 3
  - Upload response time vs start p response = 4
  - Download response time vs upload response time =3
  - Download response time vs better help = 2
  - Download response time vs better security = 2
  - Upload response time vs better help = 1
  - better security vs start up response = 4
  - better security vs upload response time = 2
- In your group discuss and evaluate the priority of the following requirements, using Priority Vector Concept.

Create a Symmetric Identity Matrix

	Start up response time	Download response time	Upload response time	Better help	better security
Start up response time	1	1/3	1/4	1/3	1/4
Download response time	3	1	3	2	2
Upload response time	4	1/3	1	1/2	1
Better Help	3	1/2	2	1	3
better security	4	1/2	1	1/3	1
Sum	**15**	**8/3**	**29/4**	**25/6**	**29/4**

Normalize the matrix

	Start up response time	Download response time	Upload response time	Better help	better security
Start up response time	1/15	3/24	4/116	6/75	4/116
Download response time	3/15	3/8	12/29	12/25	8/29
Upload response time	4/15	3/24	4/29	6/50	4/29
Better Help	3/15	3/16	8/29	6/25	12/29
better security	4/15	3/16	4/29	6/75	4/29
Sum	**1**	**1**	**1**	**1**	**1**

Priority Indexes

	Start up response time	Download response time	Upload response time	Better help	better security	Priority Indices
Start up response time	0.06	.125	.034	.08	0.34	0.12
Download response time	0.18	.375	0.414	.48	.27	0.34
Upload response time	0.26	.125	0.145	.12	0.145	0.15
Better Help	0.2	.187	0.272	.24	0.41	0.26
better security	.26	.187	0.135	.08	.148	0.16

- Download response time: 34%
- Better help: 26%
- better security: 16%
- Upload response: 15%
- Start up: 12%

References

- A.J. Bagnall, V.J. Rayward-Smith, and J.M. Whittley, “The Next Release Problem,” Information and Software Technology, vol. 43, no. 14, 2001, pp. 883–890.
- H.-W. Jung, “Optimizing Value and Cost in Requirements Analysis,” IEEE Software, vol. 15, no. 4, 1998, pp. 74–78.
- G. Ruhe and M. O. Saliu, The and Science of Software Release Planning, IEEE Software, 2005.
- J. Momoh and G. Ruhe, Release Planning Process Improvement − An Industrial Case Study, John Wiley & Sons Ltd, 2006.