---------- Historical development ----------
There is not too much to note for the historical development for this language, March 4, 2020 is the earliest available version of the langauge (1.3.2).
Since then 1.3.3 and 1.3.4 have been released, notably covering python2 support and bug fixes respectively. You can view release history here: https://github.com/aaronjanse/asciidots/releases

---------- Language overview ----------
ASCII Dots introduces a novel approach to programming, employing ASCII art paths and dots as fundamental elements of computation.
Dots serve as information carriers, traversing paths to undergo operations and control flow.
Its design emphasizes simplicity and visual clarity, making it accessible to both programmers and non-programmers alike.

---------- Language features ----------
3.1 Syntax:
ASCII Dots syntax revolves around the representation of dots and paths.
Dots are denoted by periods (`.`) or bullet symbols (`•`), serving as starting points for program execution.
Paths, depicted by vertical pipes (`|`) and horizontal dashes (`-`), guide the movement of dots.
Special paths, such as `>`, `<`, `^`, and `v`, offer additional functionality like directional output and reflection.

3.2 Data Types:
In ASCII Dots, dots function as both data carriers and computational units. They possess values and IDs, allowing for the manipulation of data along paths.
The language prioritizes simplicity by avoiding complex data structures, relying instead on the straightforward interaction between dots and paths.

3.3 Primitive Operations:
Primitive operations in ASCII Dots include basic arithmetic (addition, subtraction, multiplication, division), modulus, exponentiation, and boolean operations.
These operations are performed by dots as they traverse paths and encounter operator symbols.

3.4 Sequence Control:
Control flow in ASCII Dots is managed through the use of special paths and characters like tilde (`~`) for redirection and conditional branching.
This allows for the creation of branching logic and loops within ASCII art representations.

3.5 Programming Environment:
ASCII Dots provides an interactive console for input and output operations. Output is facilitated by the `$` symbol, with support for ASCII and newline control.
Input is prompted by the `?` symbol, allowing for user interaction within the program flow.

---------- Evaluation ----------
ASCII Dots emerged from a convergence of influences, drawing inspiration from various sources such as ASCII art, Befunge, and mechanical computers.
Its creation was motivated by a desire to blend aesthetics with functionality, offering a visually engaging programming experience unlike traditional languages.
Milestones in its development reflect a gradual refinement of its syntax and features to align with its unique design philosophy.

The language offers a unique programming experience characterized by its aesthetic appeal and simplicity.

Its visual nature makes it intuitive for certain types of tasks, particularly those involving pattern recognition and visual manipulation.
However, its esoteric nature and unconventional syntax may pose challenges for developers accustomed to more traditional languages.
Overall, ASCII Dots represents a creative exploration of programming concepts through artistic expression, albeit with limitations in practical applicability for general-purpose programming tasks.

--------------- Bibliograohy ---------------
repository----------------https://github.com/daleUrquhart/ASCII-Dots-Project?tab=readme-ov-file
documentation-------------https://ajanse.me/asciidots/language/
library documentation-----https://ddorn.github.io/asciidots/docs/libs/
eso-langs article---------https://esolangs.org/wiki/AsciiDots
presentation--------------https://github.com/daleUrquhart/ASCII-Dots-Project

--------------- Appendix ---------------
Sample programs can be found below
https://github.com/daleUrquhart/ASCII-Dots-Project/tree/main/Sample%20Programs