Monads are important programming tools in functional languages. They allow the programmer to build up computations using sequential building blocks. However, it is observed in practice that monadic programs fail to modularize crosscutting concerns properly. AspectH, an aspect oriented extension of Haskell, was developed to offer new means of separation of concerns in monadic programs. This paper presents a case study where we use AspectH to redesign the weaver that implements the AspectH language itself. The analysis of the results includes a discussion about the strategy of introducing monads only in late stages of the development of functional programs, used in the development of the AspectHs weaver.