

```
(defun parse-log-entry (text)
  (if (well-formed-log-entry-p text)
      (make-instance 'log-entry ...)
      (error 'malformed-log-entry-error :text text)))
```

```
(defun parse-log-file (file)
  (with-open-file (in file :direction :input)
    (loop for text = (read-line in nil nil) while text
          for entry = (handler-case (parse-log-entry text)
                                   (malformed-log-entry-error () nil))
                    when entry collect it)))
```

Does too much!

Restarts

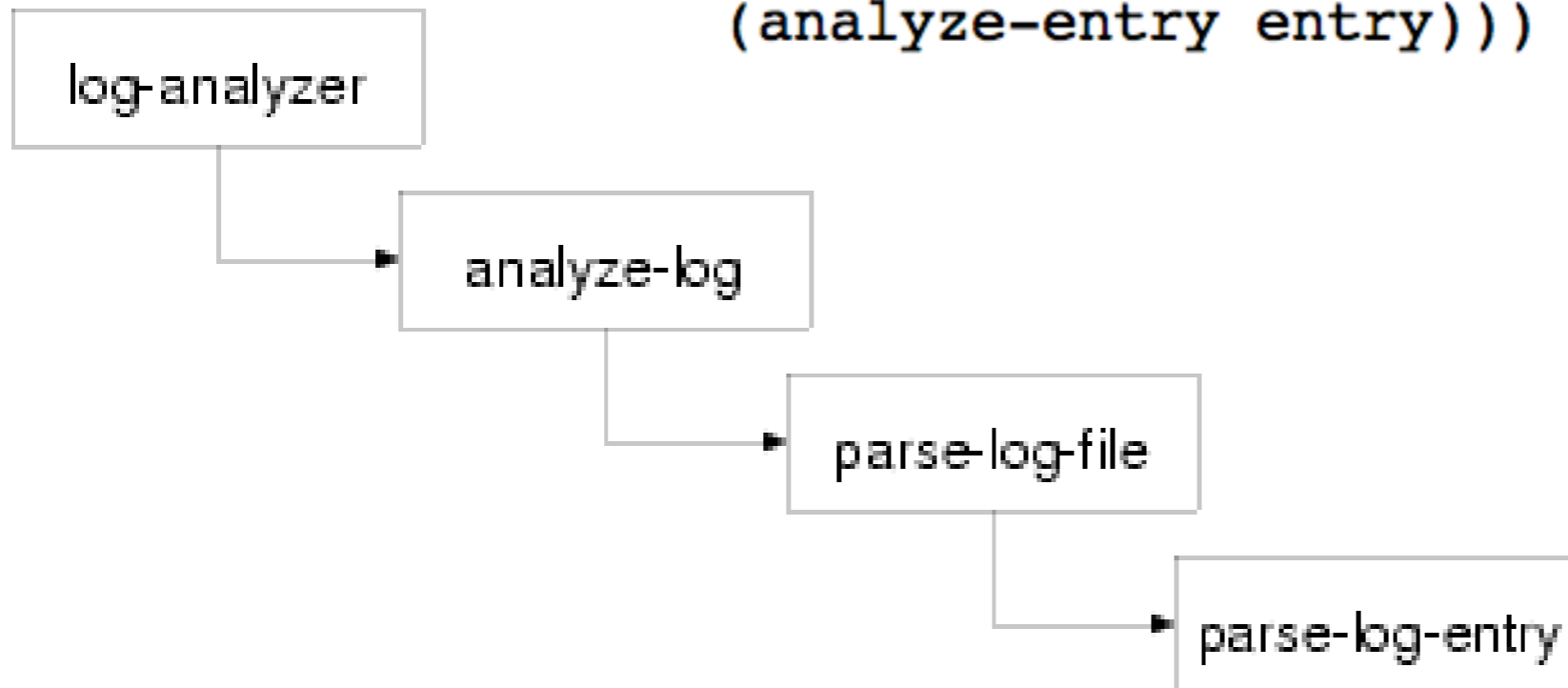
```
(defun parse-log-file (file)
  (with-open-file (in file :direction :input)
    (loop for text = (read-line in nil nil) while text
          for entry = (restart-case (parse-log-entry text)
                                   (skip-log-entry () nil))
                    when entry collect it)))
```

Application: log-analyzer

```
(defun log-analyzer ()  
  (dolist (log (find-all-logs))  
    (analyze-log log)))
```

No error handling

```
(defun analyze-log (log)  
  (dolist (entry (parse-log-file log))  
    (analyze-entry entry)))
```



With error handling

```
(defun log-analyzer ()  
  (handler-bind ((malformed-log-entry-error  
                  #'(lambda (c)  
                      (invoke-restart 'skip-log-entry))))  
    (dolist (log (find-all-logs))  
      (analyze-log log))))
```

```
(defun analyze-log (log)  
  (dolist (entry (parse-log-file log))  
    (analyze-entry entry)))
```

Multiple restarts

In the log parsing library:

```
(defun parse-log-entry (text)
  (if (well-formed-log-entry-p text)
      (make-instance 'log-entry ...)
      (restart-case (error 'malformed-log-entry-error :text text)
                    (use-value (value) value)
                    (reparse-entry (fixed-text) (parse-log-entry fixed-text)))))
```

In the application:

```
(defun log-analyzer ()
  (handler-bind ((malformed-log-entry-error
                  #'(lambda (c)
                      (use-value
                       (make-instance 'malformed-log-entry :text (text c))))))
    (dolist (log (find-all-logs))
      (analyze-log log))))
```