```
static_diagram DIGITALVOTERLIST
component
  cluster STATIONANDMANAGER
  component
    deferred class STATION
      feature
        Address : IPADDRESS

        Manager : IPADDRESS

        SetManager : void
          -> address : IPADDRESS
          require address /= void
          ensure Manager = address
        end

        EnoughStations : BOOLEAN

        ElectionInProgress : BOOLEAN

        Peers : SORTED_LIST[IPADDRESS]

        Database : IDATABASE

        Communicator : ICOMMUNICATOR

        Crypto : ICRYPTO

        SetCrypto : void
          -> newcrypto : ICRYPTO
          require newcrypto /= void
          ensure Crypto = newcrypto
        end

        Logger : ILOGGER

        SetLogger : void
          -> newlogger : ILOGGER
          require newlogger /= void
          ensure Logger = newlogger
        end

        UI : IDVLUI

        IsManager : BOOLEAN

        Listening : BOOLEAN

        MasterPassword : VALUE

        SetMasterPassword : void
          -> password : VALUE
          require password /= void and MasterPassword = void
          ensure MasterPassword = password
        end

        StationActive : BOOLEAN
          -> address : IPADDRESS
          require address /= void
        end

        DiscoverPeers : SEQUENCE[IPADDRESS]
```

```
            ensure result /= void
          end

          ValidMasterPassword : BOOLEAN
            -> password : STRING
            require password /= void
          end

          ShutDownElection : void

          ExchangePublicKeys : void
            -> address : IPADDRESS
            require address /= void and StationActive(address)
          end

          StartListening : void
            require not Listening
            ensure Listening
          end

          StopListening : void
            require Listening
            ensure not Listening
          end

          StartElection : void
            require not ElectionInProgress
            ensure ElectionInProgress
          end

          EndElection : void
            require ElectionInProgress
            ensure not ElectionInProgress
          end

          AddPeer : void
            -> address : IPADDRESS
            -> key : ASYMMETRICKEY
            require address /= void and not Peers.Contains(address)
            ensure Peers.Contains(address)
          end

          RemovePeer : void
            -> address : IPADDRESS
            require address /= void and Peers.Contains(address)
            ensure not Peers.Contains(address)
          end

          StartNewManagerElection : void

          ElectNewManager : void
            require not StationActive(Manager)
            ensure Manager /= old Manager
          end

          RequestBallot : void
            -> voterNumber : VOTERNUMBER
            -> cpr : CPR
            require Database.get(voterNumber, cpr) = NOTRECEIVED
          end

          RequestBallotCPROnly : void
```

```
            -> cpr : CPR
            -> password : STRING
            require password /= void and ValidMasterPassword(password) and Database.get(cpr, password) = NOTRECEIVED
          end

          BallotReceived : void
            -> voterNumber : VOTERNUMBER
            -> cpr : CPR
            require Database.get(voterNumber, cpr) = NOTRECEIVED
            ensure Database.get(voterNumber, cpr) = RECEIVED
          end

          BallotReceivedCPROnly : void
            -> cpr : CPR
            -> password : STRING
            require password /= void and ValidMasterPassword(password) and Database.get(cpr, password) = NOTRECEIVED
            ensure Database.get(cpr, password) = RECEIVED
          end

          RevokeBallot : void
            -> voterNumber : VOTERNUMBER
            -> cpr : CPR
            require Database.get(voterNumber, cpr) = RECEIVED
            ensure Database.get(voterNumber, cpr) = NOTRECEIVED
          end

          RevokeBallotCPROnly : void
            -> cpr : CPR
            -> password : STRING
            require password /= void and ValidMasterPassword(password) and Database.get(cpr, password) = RECEIVED
            ensure Database.get(cpr, password) = NOTRECEIVED
          end

          AnnounceAddPeer : void
            -> newPeerAddress : IPADDRESS
            -> newPeerKey : ASYMMETRICKEY
            require IsManager and newPeerAddres /= void
          end

          AnnounceRemovePeer : void
            -> removePeerAddress : IPADDRESS
            require IsManager and removePeerAddress /= void
          end

          PromoteNewManager : void
            -> newManagerAddress : IPAddress
            require IsManager and newManagerAddress /= void
          end

          AnnounceStartElection : void
            require IsManager and not ElectionInProgress
            ensure ElectionInProgress
          end

          AnnounceEndElection : void
            require IsManager and ElectionInProgress
            ensure not ElectionInProgress
          end

          invariant
            Address /= void and Peers /= void
      end
    end
  end
```