In [1]:

```
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.patches as mpatches
```

In [2]:

```
# 1. Generate structured synthetic data
# 100 samples, 10 features (6 informative)(2 useless)(2 Noicy)

data, labels = make_classification(n_samples = 100, n_features = 10,
                                   n_informative= 6, n_redundant = 2,
                                   n_classes =2, random_state =42)
```

In [3]:

```
# 2. Split data into training and testing sets

X_train, X_test, y_train,y_test = train_test_split(data, labels, test_size =0.2, random_state = 42)
```

In [4]:

```
# 3. Standardize features

scaler = StandardScaler()
X_trained_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [5]:

```python
# 4. Define AIRS-like training with SVM

def ais_training(X_train, y_train, num_detectors = 20):

    # Randomly select samples as detectors
    detector_indices = np.random.choice(len(X_train), num_detectors,replace = False)
    detectors = X_train[detector_indices]

    # Train SVM classifier on detectors
    svm = SVC(kernel ='rbf', probability = True)
    svm.fit(detectors , y_train[detector_indices])


    return svm
```

In [6]:

```python
print(data)
```

```
  -2.09353641e+00 -8.62288841e-01  4.07685401e+00  1.19622401e+00
  -5.60426714e-01 -5.64078631e-01]
 [-2.31529492e+00 -1.71815085e+00 -1.03128854e-01  2.87448229e-02
  -1.45688196e+00  6.88691737e-01 -7.77405288e-01 -3.12325566e+00
   2.83647496e-01  1.27845186e+00]
 [ 7.99623146e-01  1.95297150e+00  3.52798052e+00  9.61207769e-02
  -2.55874817e-01 -3.04053667e+00 -5.44574220e-01  4.42299896e+00
  -1.83107165e+00 -4.62275289e-01]
 [-8.88711534e-01 -6.13853842e-01  1.91673375e+00  1.54750520e+00
  -1.44022214e+00 -1.57429748e+00 -1.67216684e+00  6.41999941e-01
  -1.53368231e+00  1.79587767e+00]
 [ 3.67232221e-01  8.53508003e-01  3.23005438e+00  4.40014450e-01
  -2.50398440e+00 -2.33804997e+00 -2.05791564e-01  1.56385556e+00
  -2.91981779e-01 -5.02054224e-01]
 [ 2.12577537e+00 -7.57130138e-01 -1.14939998e+00 -3.95551539e-02
  -7.36464840e-01 -1.13706280e+00  6.52718169e-01  6.81698643e-01
   1.92142556e+00  6.81500697e-01]
 [-3.53725096e+00 -2.70125390e+00  5.65655032e-01  6.45484181e-01
  -4.30295872e+00  1.83665458e+00 -1.81489985e+00 -6.65690540e+00
   6.11408486e-01  2.16325472e+00]
```

In [7]:
```
print(labels)
```

```
[1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1
 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 1 0 1
 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0]
```

In [8]:
```
# 5. Train AIRS-inspired model
svm_classifier = ais_training(X_trained_scaled, y_train,num_detectors = 30 )
```

In [9]:
```
prediction = svm_classifier.predict(X_test_scaled)

print("Classification Report:")
print(classification_report(y_test, prediction))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.80      0.76        10
           1       0.78      0.70      0.74        10

    accuracy                           0.75        20
   macro avg       0.75      0.75      0.75        20
weighted avg       0.75      0.75      0.75        20
```

In [10]:
```
print(accuracy_score(y_test, prediction) *100)
```

```
75.0
```

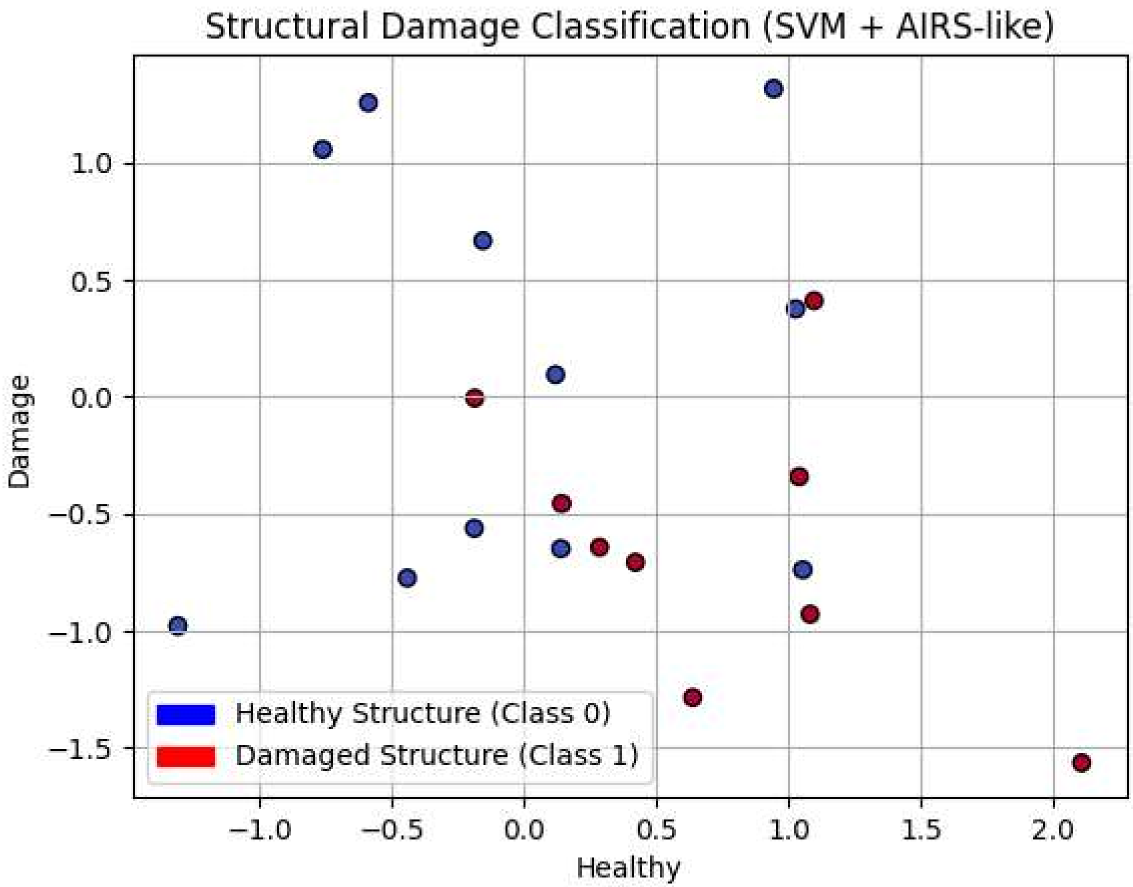
In [11]:
```
accuracy = np.mean(prediction == y_test)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 75.00%
```

In [12]:

```
#Visualize (using first 2 features)

healthy_patch = mpatches.Patch(color='blue', label='Healthy Structure (Class 0)')
damaged_patch = mpatches.Patch(color='red', label='Damaged Structure (Class 1)')
plt.legend(handles=[healthy_patch, damaged_patch])
plt.scatter(X_test_scaled[:, 0], X_test_scaled[:, 1], c=prediction, cmap='coolwarm', edgecolor='k')
plt.title("Structural Damage Classification (SVM + AIRS-like)")
plt.xlabel("Healthy")
plt.ylabel("Damage")
plt.grid(True)
plt.show()
```



In [ ]: