```
# Pseudo-Python code for imputing missing attributes using ART-DWD model

# Inputs:
# X : vector of m attributes (some may be missing)
# I_X : set of indices of missing attributes in X
# P_a : threshold for maximum allowable missing attributes
# P_r : similarity threshold for recognition
# ART_WD_model : trained ART-WD network

def impute_missing_ARTWD(X, I_X, P_a, P_r, ART_WD_model):
    q = len(I_X)  # number of missing attributes

    # Step 2: Check if recovery is possible
    if q > P_a:
        return "Recovery not possible"

    # Step 4: Reduce input vector by removing missing attributes
    X_reduced = [X[i] for i in range(len(X)) if i not in I_X]
    z = len(X_reduced)  # reduced dimension

    # Step 5: Build reduced ART-WD model
    reduced_model = ART_WD_model.copy()

    # 5.1: Remove input neurons corresponding to missing attributes
    for g in I_X:
        reduced_model.remove_input_neuron(g)

    # 5.2: Remove descending synaptic connections from recognition neurons to missing attributes
    for g in I_X:
        reduced_model.remove_descending_connections(g)

    # Step 6: Feed reduced vector into reduced network
    k_star = reduced_model.num_classes
    reduced_model.feed_input(X_reduced)

    # Step 7: Find winner neuron in recognition layer
    j_star = reduced_model.find_winner_neuron()

    # Step 8: Calculate similarity measure RN
    RN = reduced_model.calculate_similarity(j_star)

    # Step 9: Check similarity
    if RN >= P_r:
        if k_star > 1 and reduced_model.output[j_star] < P_r:
            reduced_model.remove_class(j_star)
            k_star -= 1
            # Repeat Step 7
            j_star = reduced_model.find_winner_neuron()
            RN = reduced_model.calculate_similarity(j_star)
        else:
            return "Recovery not possible"

    # Step 10: Recover missing attributes
    for g in I_X:
        X[g] = sum([vec[g] for vec in reduced_model.get_class_vectors(j_star)]) / len(reduced_model.get_class_vectors(j_star))

    return X
```