Models for complex system behaviour of natural hazards in mountainous regions

1. Supplementary material

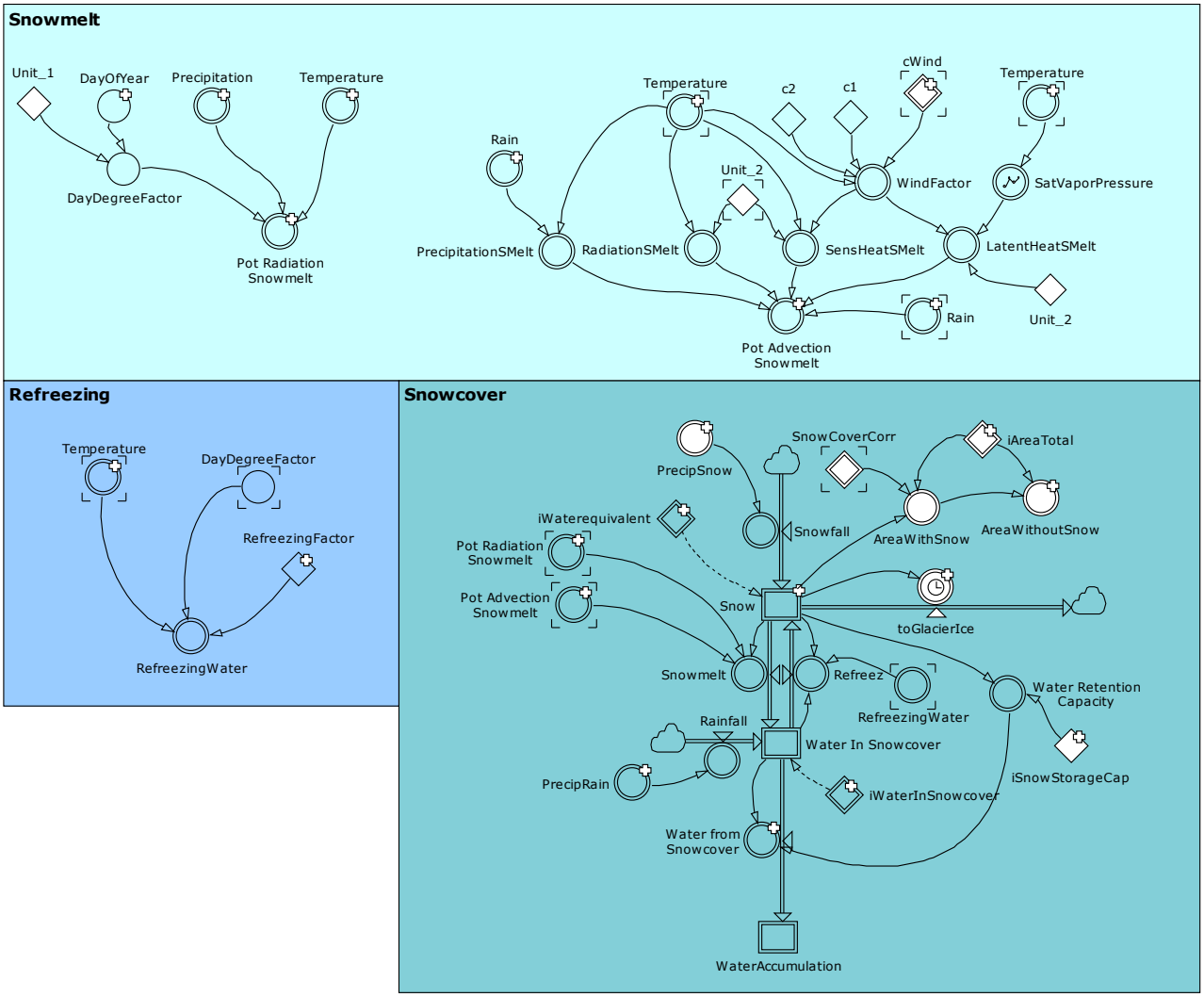

Fig. 1 Example of a system model on snowpack already developed

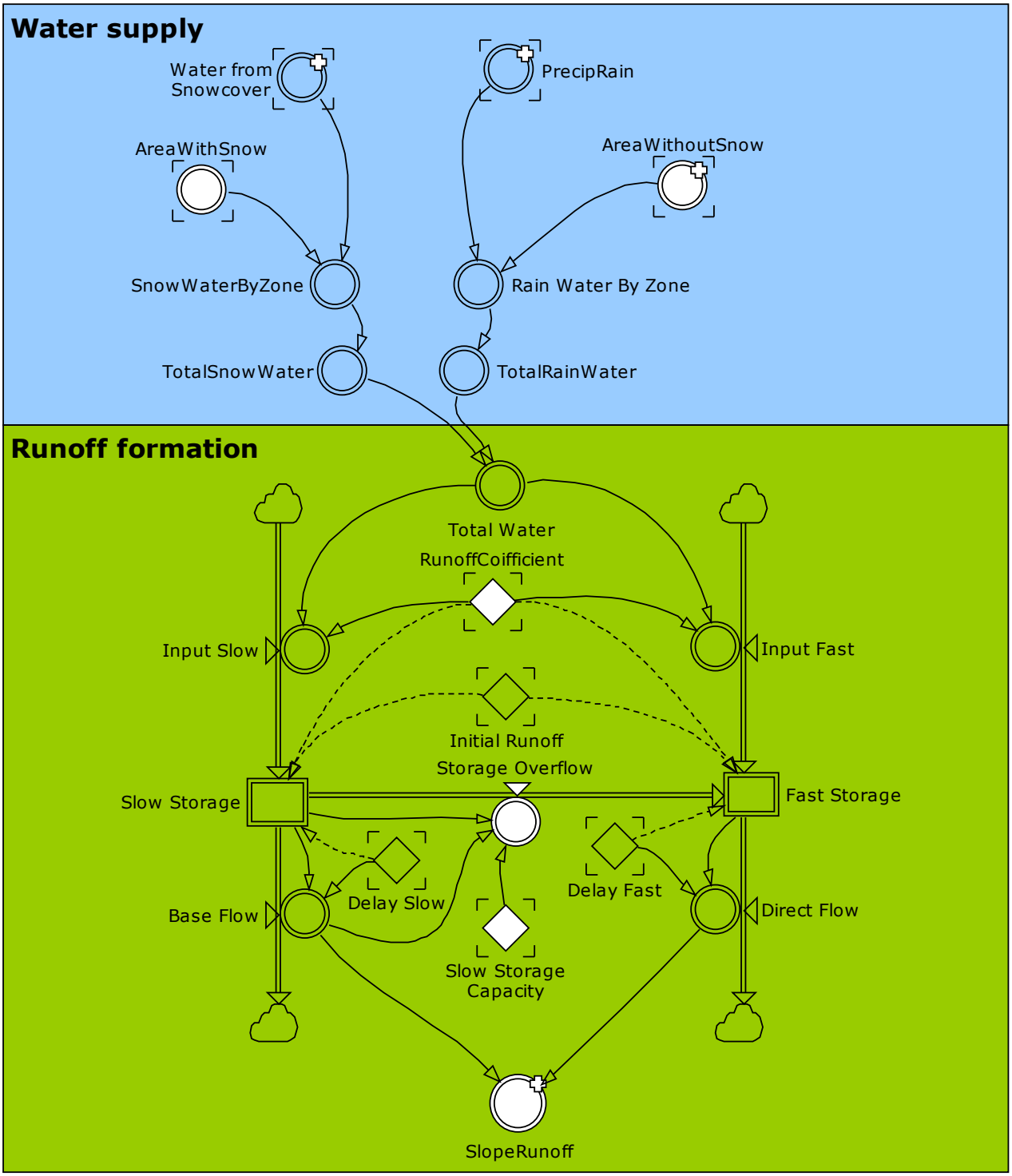

Fig. 2 Example of a system model on runoff formation already developed

```
mainmodel Bächlital {
    const AreaGlacier {
        autotype Real
        unit ha
        dim Altitude
        init Parent~iAreaGlacier[*;'012Bächlital']
    }
    aux AreaWithoutSnow {
        autotype Real
        unit ha
        dim Ensemble; Altitude
        def FOR (i=Ensemble; j=Altitude |
                iAreaTotal[j]#-AreaWithSnow[i;j])
    }
    aux AreaWithSnow {
        autotype Real
        unit ha
        dim Ensemble; Altitude
        def FOR (i=Ensemble; j=Altitude |
                IF(Snow[i;j] > SnowCoverCorr[j]; iAreaTotal[j]; iAreaTotal[j] #*(Snow[i;j]/SnowCoverCorr[j])))
    }
    aux Base Flow {
        autotype Real
        unit m³/s
        autodim Ensemble
        def 'Slow Storage'/'Delay Slow'
    }
    const c1 {
        autotype Real
        init 0
        doc Meltingfactor depending on temperature
    }
    const c2 {
        autotype Real
        init .5
        doc Meltingfactor depending on wind
    }
    const CapacityOberaarsee {
        autotype Real
        unit m³
        init 57300000
    }
    const cWind {
        autotype Real
        autodim Altitude
        init Parent~cWind
        doc Unit: m/s
            Is not used as the formula needs unitless parameter
    }
    aux DayDegreeFactor {
        autotype Real
        autounit mm/(da*C)
        def (SIN((DayOfYear-81)/365*2*PI)+3.5)*Unit_1
    }
    const DayDegreeGlacier {
        autotype Real
        unit mm/C/da
        init Parent~GlacierDayDegree
    }
    aux DayOfYear {
        type Integer
        def Parent~'ClimateSzenario Grimsel'.oDayOfYear
    }
    const Delay Fast {
        autotype Real
        unit hr
        init 2
```

```
}
const Delay Slow {
    autotype Real
    unit da
    init 6
}
aux Direct Flow {
    autotype Real
    unit m³/s
    autodim Ensemble
    def MIN('Fast Storage'/'Delay Fast';'Fast Storage'/TIMESTEP)
}
level Fast Storage {
    autotype Real
    autounit m³
    dim Ensemble
    init 'Initial Runoff'*RunoffCoifficient*'Delay Fast'
    inflow { autodef  'Input Fast' }
    outflow { autodef  'Direct Flow' }
    inflow { autodef  'Storage Overflow' }
}
aux GlacierMelt {
    autotype Real
    unit mm/da
    dim Ensemble; Altitude
    def FOR (i=Ensemble; j=Altitude |
          IF (iAreaTotal[j] > 0 <<m²>> AND Temperature[i;j] > 0<<C>>; Temperature[i;j]*DayDegreeGlacier*
       (AreaWithoutSnow[i;j]/iAreaTotal[j]); 0<<mm/da>>))
}
aux GlacierRunoff {
    autotype Real
    unit m³/s
    dim Ensemble
    def ARRSUM(WaterFromGlacierByZone; 2; 2)
}
const iAreaTotal {
    autotype Real
    unit ha
    dim Altitude
    init Parent~iAreaWatershed[*;'012Bächlital']
}
aux Inflow {
    autotype Real
    autounit m3/s
    dim Ensemble
    def SlopeRunoff+GlacierRunoff
}
const Initial Runoff {
    autotype Real
    unit m^3/s
    init 1
}
const InitOberaarsee {
    autotype Real
    init 0.8
}
aux Input Fast {
    autotype Real
    autounit m3/s
    autodim Ensemble
    def 'Total Water'*RunoffCoifficient
}
aux Input Slow {
    autotype Real
    autounit m3/s
    autodim Ensemble
    def ('Total Water'*(1-RunoffCoifficient))
```

```
}
level IntakeBächli {
    autotype Real
    unit m³
    dim Ensemble
    init 0
    outflow { autodef  toRäBoSee }
    outflow { autodef  toGrimselsee }
    inflow { autodef  Inflow }
    inflow { autodef  Parent~Gruben.toBächlital }
}
const iSnowStorageCap {
    autotype Real
    init Parent~SnowStorageCap
}
const iWaterequivalent {
    autotype Real
    unit mm
    dim Altitude
    init Parent~iWaterEquivalent
}
const iWaterInSnowcover {
    autotype Real
    unit mm
    dim Altitude
    init Parent~iWaterInSnowcover
}
aux LatentHeatSMelt {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def (WindFactor*(SatVaporPressure-6.11<<C>>)/0.67)*Unit_2
    doc Schulla [2.64], 0.67 = psychrometer constant
}
aux Pot Advection Snowmelt {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def FOR(i=Ensemble; j=Altitude |
          IF(Rain[i;j]>=5<<mm/da>>; PrecipitationSMelt[i;j]+RadiationSMelt[i;j]+LatentHeatSMelt[i;j]+
        SensHeatSMelt[i;j]; 0<<mm/da>>)
        )
}
aux Pot Radiation  Snowmelt {
    autotype Real
    autounit mm/da
    dim Ensemble;Altitude
    def FOR(i=Ensemble; j=Altitude |
          IF(Precipitation[i]<5.0<<mm/da>>;IF(Temperature[i;j]>0<<C>>; (DayDegreeFactor*Temperature[i;j]);
        0<<mm/da>>);0<<mm/da>>)
        )
}
aux Precipitation {
    autotype Real
    unit mm/da
    autodim Ensemble
    def Parent~'ClimateSzenario Grimsel'.Ens_Precip1_GRH
}
aux PrecipitationSMelt {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def FOR(i=Ensemble; j=Altitude |
          IF(Temperature[i;j]>0<<C>> ;(0.0125<<1/C>>*Rain[i;j]*Temperature[i;j]);0<<mm/da>>)
        )
}
aux PrecipRain {
```

```
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def Parent~'ClimateSzenario Grimsel'.oRain_GRH
}
aux PrecipSnow {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def Parent~'ClimateSzenario Grimsel'.oSnow_GRH
}
aux RadiationSMelt {
    autotype Real
    autounit mm/da
    dim Ensemble;Altitude
    def FOR(i=Ensemble; j=Altitude |
          IF(Temperature[i;j]>0<<C>>;(Temperature[i;j]*1.2*Unit_2);0<<mm/da>>))
}
aux Rain {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def Parent~'ClimateSzenario Grimsel'.oRain_GRH
}
aux Rain Water By Zone {
    autotype Real
    unit m³/s
    dim Ensemble; Altitude
    def FOR (i=Ensemble; j=Altitude | PrecipRain[i;j]#*AreaWithoutSnow[i;j])
}
aux Rainfall {
    autotype Real
    unit mm/hr
    dim Ensemble;Altitude
    def PrecipRain
}
aux Refreez {
    autotype Real
    unit mm/hr
    dim Ensemble; Altitude
    def FOR (i=Ensemble; j=Altitude |
          IF (Snow[i;j] >10<<mm>>; MIN(RefreezingWater[i;j];'Water In Snowcover'[i;j]/TIMESTEP)))
}
const RefreezingFactor {
    autotype Real
    init Parent~RefreezingFactor
}
aux RefreezingWater {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def FOR(i=Ensemble; j=Altitude |
          IF(Temperature[i;j]<0<<C>>;-Temperature[i;j]*DayDegreeFactor*RefreezingFactor;0<<mm/da>>))
}
const RunoffCoifficient {
    autotype Real
    init .5
}
aux SatVaporPressure {
    autotype Real
    autounit C
    dim Ensemble;Altitude
    def FOR(i=Ensemble; j=Altitude |
          GRAPH(Temperature[i;j];0<<C>>;5<<C>>;{6.11;8.7;12.2;17;23.3;31.6//Min:0;Max:40//}<<C>>)
        )
}
aux SensHeatSMelt {
```

```
    autotype Real
    autounit mm/da
    dim Ensemble;Altitude
    def WindFactor*Temperature*Unit_2
}
aux SlopeRunoff {
    autotype Real
    unit m^3/s
    autodim Ensemble
    def 'Base Flow'+'Direct Flow'
}
level Slow Storage {
    autotype Real
    autounit m³
    dim Ensemble
    init 'Initial Runoff'*(1-RunoffCoifficient)*'Delay Slow'*2.5
    inflow { autodef  'Input Slow' }
    outflow { autodef  'Base Flow' }
    outflow { autodef  'Storage Overflow' }
}
const Slow Storage Capacity {
    autotype Real
    unit m³
    init 7000000
}
level Snow {
    type Real
    unit mm
    dim Ensemble; Altitude
    init FOR (i=Ensemble; j=Altitude|iWaterequivalent[j])
    inflow { autodef  Refreez }
    outflow { autodef  Snowmelt }
    inflow { autodef  Snowfall }
    outflow { autodef  toGlacierIce }
}
const SnowCoverCorr {
    autotype Real
    unit mm
    dim Altitude
    init {100;100;100;200;200;200;300;300;500;500;1000;1500;2000;2500;3000;3500;4000;4500}
}
aux Snowfall {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def PrecipSnow
}
aux Snowmelt {
    autotype Real
    autounit mm/da
    dim Ensemble; Altitude
    def MIN(('Pot Radiation  Snowmelt'+'Pot Advection Snowmelt');Snow/TIMESTEP)
    doc Geändert, da Snow nun als Reservoir definiert und daher keine MIN-Abfrage mehr notwendig
}
aux SnowWaterByZone {
    autotype Real
    unit m³/s
    dim Ensemble;Altitude
    def FOR (i=Ensemble; j=Altitude | 'Water from Snowcover'[i;j]#*AreaWithSnow[i;j])
}
aux Storage Overflow {
    autotype Real
    unit m³/s
    autodim Ensemble
    def MAX(IF('Slow Storage'>'Slow Storage Capacity';(('Slow Storage'-'Slow Storage Capacity')/TIMESTEP)
        -'Base Flow';0<<m³/s>>);0<<m³/s>>)
}
```

```
const SwitchToGrimsel {
    autotype Real
    init 1
}
aux Temperature {
    autotype Real
    unit C
    dim Ensemble;Altitude
    def Parent~'ClimateSzenario Grimsel'.Ens_Temperature
}
aux toGlacierIce {
    autotype Real
    autounit mm/hr
    dim Ensemble; Altitude
    def FOR (i=Ensemble; j=Altitude | PULSE(Snow[i;j];STARTTIME+365 <<da>>;365 <<da>>))
    firstorder
}
aux toGrimselsee {
    autotype Real
    autounit m3/s
    autodim Ensemble
    def IF (SwitchToGrimsel = 1; IntakeBächli/TIMESTEP;0 <<m³/s>>)
}
aux toRäBoSee {
    autotype Real
    autounit m3/s
    autodim Ensemble
    def IF (SwitchToGrimsel=0; IntakeBächli/TIMESTEP; 0 <<m³/s>>)
}
aux Total Water {
    autotype Real
    unit m³/s
    autodim Ensemble
    def TotalSnowWater+TotalRainWater
}
aux TotalRainWater {
    autotype Real
    unit m³/s
    autodim Ensemble
    def ARRSUM('Rain Water By Zone';2 ;2)
}
aux TotalSnowWater {
    autotype Real
    unit m³/s
    autodim Ensemble
    def ARRSUM(SnowWaterByZone; 2; 2)
}
const TurbinCapacityG1 {
    autotype Real
    unit m³/s
    init 7.5
}
const TurbinCapacityG2 {
    autotype Real
    unit m³/s
    init 93
}
const Unit_1 {
    autotype Real
    unit mm/(C*da)
    init 1
}
const Unit_2 {
    autotype Real
    unit mm/(C*da)
    init 1
}
```

```
    aux WARcomponent {
        autotype Real
        unit m³/da
        dim WARcomponent;Ensemble
        def VECTOR(TotalRainWater;TotalSnowWater;GlacierRunoff)
    }
    aux Water from Snowcover {
        autotype Real
        unit mm/hr
        dim Ensemble;Altitude
        def FOR(i=Ensemble; j=Altitude |
              (MAX('Water In Snowcover'[i;j]-'Water Retention Capacity'[i;j];0<<mm>>))/TIMESTEP)
    }
    level Water In Snowcover {
        autotype Real
        unit mm
        dim Ensemble; Altitude
        init FOR (i=Ensemble; j=Altitude | iWaterInSnowcover[j])
        outflow { autodef  Refreez }
        inflow { autodef  Rainfall }
        outflow { autodef  'Water from Snowcover' }
        inflow { autodef  Snowmelt }
    }
    aux Water Retention Capacity {
        autotype Real
        autounit mm
        dim Ensemble; Altitude
        def Snow*iSnowStorageCap
    }
    level WaterAccumulation {
        autotype Real
        unit mm
        dim Ensemble;Altitude
        init 0
        inflow { autodef  'Water from Snowcover' }
    }
    aux WaterFromGlacierByZone {
        autotype Real
        autounit m3/s
        dim Ensemble; Altitude
        def FOR (i=Ensemble; j=Altitude |
              GlacierMelt[i;j]#*AreaGlacier[j])
    }
    aux WindFactor {
        autotype Real
        dim Ensemble;Altitude
        def FOR(i=Ensemble; j=Altitude |
              IF(Temperature[i;j]>0<<C>>;(c1+(c2*cWind[j]));0)
            )
    }
}
range Altitude {
    def H0500; H0700; H0900; H1100; H1300; H1500; H1700; H1900; H2100; H2300; H2500; H2700; H2900;
        H3100; H3300; H3500; H3700; H3900
}
range Ensemble {
    def ETHZ_HadCM3; HC_HadCM3; SMHI_HadCM3; SMHI_ECHAM5; MPI_ECHAM5; KNMI_ECHAM5;
        ICTP_ECHAM5; DMI_ECHAM5; CNRM_ARPEGE; CMHI_BCM
}
range EZG {
    def '010Oberaarsee'; '011Grimselsee'; '012Bächlital'; '013RäBoSee'; '014Gruben'; '015Handegg';
        '016Gelmersee'; '017Mattenalp'; '020Triftsee'; '021uTrift'; '022Steingletscher'; '023Stein'; '024Wendengl';
        '025Wenden'; '026Engstlenalp'; '101Gadmerwasser'; '102AareInnertk'; '103AareBrienzwiler'
}
range Scenario {
    def REFE; '2060'; '2085'
}
```

```
range WARcomponent {
    def Rain; Snow; Glacier
}
unit Aren {
    def 100m*m
    doc Are
    note Flächenmass
}
unit C {
    def @(__KELVIN; 273.15@__KELVIN)
    doc Celcius - Temperature
}
unit ha {
    def 100Aren
}
unit m {
    def __METER
    doc Meter - Length
}
unit m3 {
    def m^3
}
unit m3/s {
    def m^3/s
    doc Runoff
}
unit mm {
    def (1/1000)m
    doc Millimeter
}
```